Developer Studio Guide

1.1.0

Table of Contents

Overview

Developer Studio is a tool that streamlines the custom development process by acting as a bridge between OneStream and your local development environment. Developer Studio lets you bring OneStream artifacts to your local machine and use them with your existing development tools. Developers can also leverage IntelliSense within their Integrated Development Environment (IDE) of choice to accelerate the development process. These capabilities connect OneStream to the modern developer experience by enabling the use of AI-assisted editors in the development process.

With Developer Studio, you can:

- Sync OneStream artifacts between your local machine and your OneStream environment.
- Download reference assemblies and integrate IntelliSense capabilities.
- Efficiently manage and package OneStream objects and solutions with XF Projects.
- Leverage the tools you want, including the IDE of your choice, AI tools, and source control tools.

Setup and Installation

Before you install Developer Studio, review these details.

Dependencies

Component	Description
9.2.0	Minimum platform version required to use Developer Studio
9.4.0	Minimum Platform Version 9.4.0 is required to access full XML item configuration options, including loading and extracting XF Projects

Considerations

	Installer Deployment
Operating system	Windows
Type of install	Traditional installation. The Developer Studio application displays in the Start menu and Add or Remove programs.
End-user deployment	OneStream Solution Exchange

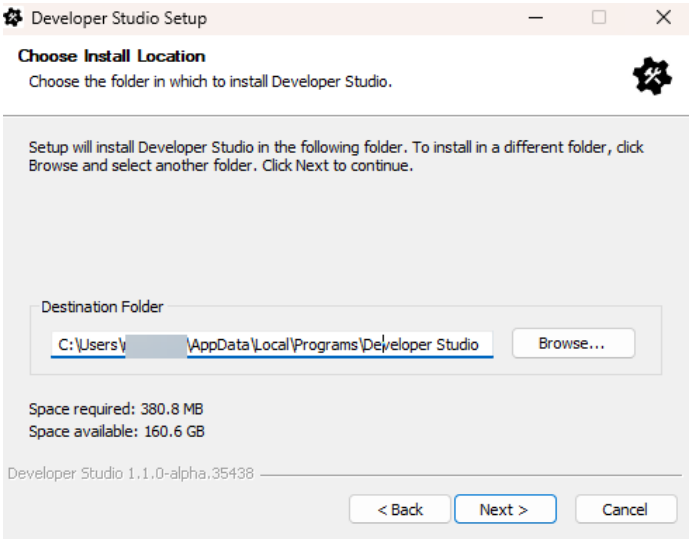
	Installer Deployment
Requires local administrator	No
Version upgrade	Manual upgrade with installer file
Use multiple versions on the same machine	No
Connect to different servers from single installed instance	Yes

Install Developer Studio

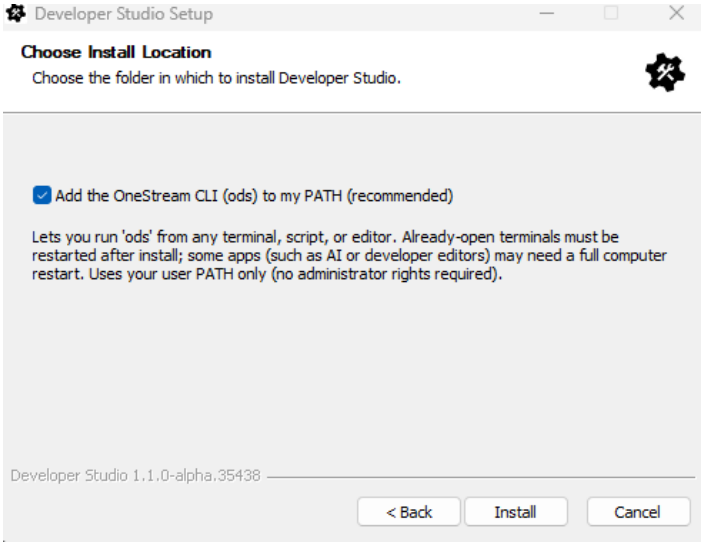
The Developer Studio installer is available on Solution Exchange. Complete the following steps to install Developer Studio:

> **NOTE:** Use the installer to upgrade Developer Studio. The installer updates the existing client without removing data.

1. Double-click the Developer Studio installer file and then select the **Next** button.
2. If necessary, change the folder path. Select the **Next** button.

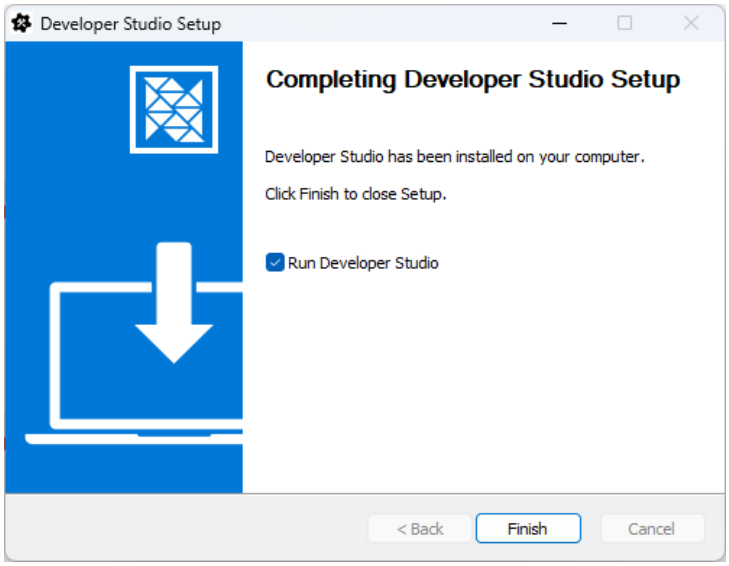


3. The Add the OneStream CLI (ods) to my PATH checkbox is selected by default. This enables you to run ods commands from any terminal, script, or editor. To disable CLI integration, deselect the checkbox. To continue, select the **Install** button.

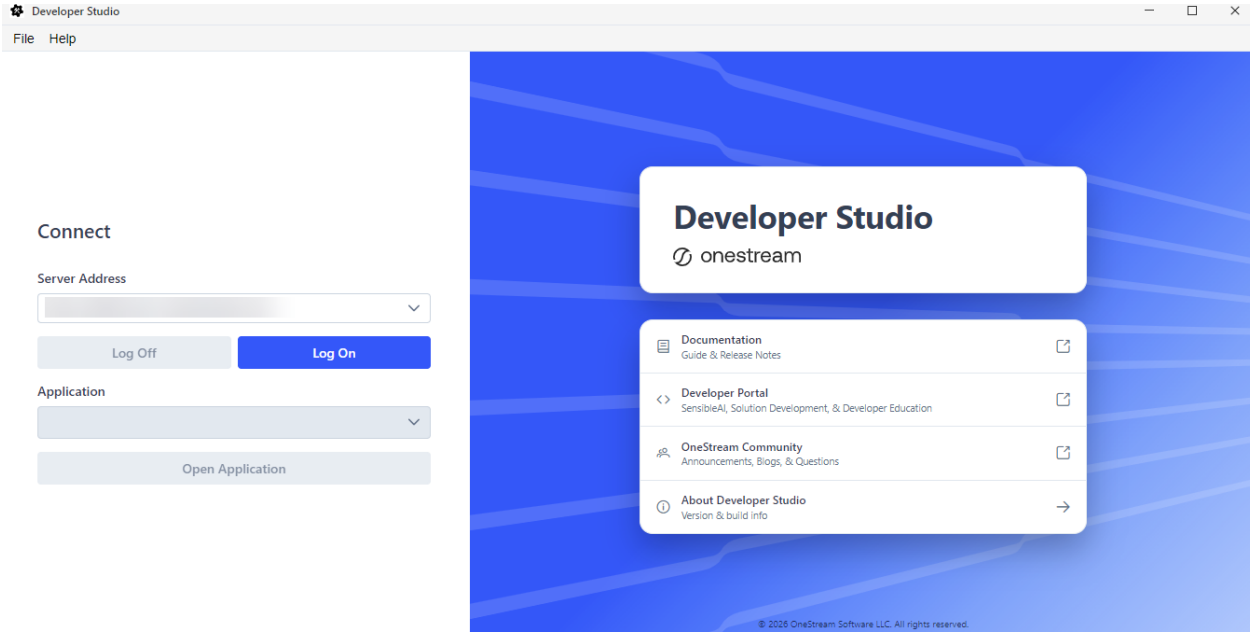


> **NOTE:** Already-opened terminals must be restarted after Developer Studio is installed for CLI integration to function. Some applications, such as AI or developer editors, may require a full computer restart.

4. Select the **Finish** button to complete the installation.



The Developer Studio launch page displays, enabling you to log in and access Developer Studio resources. See Connect



Custom Environment Configuration

> **IMPORTANT:** This section only applies if you have a custom environment configured that you would like to make compatible with Developer Studio.

> **NOTE:** OneStream Identity Server (OIS) must be installed and configured in your custom environment before completing this procedure.

When Developer Studio is installed and launched, a configuration file is created in your local AppData folder at the following path: C:\Users\<username>\.ods. To connect to custom OneStream installations, configuration changes are required to successfully complete an OIDC workflow with your local OIS setup. The configuration file provides the flexibility to define the `authorityUrl` for the server you are connecting to and enables you to disable SSL verification for self-signed certificates.

To set the `authorityUrl` for local login, open the configuration file and modify the `authorityUrl` value to match the host/port that is set up locally for the target OIS instance. You can specify multiple custom environments by adding each to the servers list.

```
{
  "defaultRepositoryRootFolder": "C:\\demoRepo",
  "lastSelectedRepositoryId": "7b903603-1d4d-410e-86e0-54f35f8000c6",
  "enableDarkMode": true,
  "showConfirmationDialogForPushAndPull": false,
  "language": "en",
  "referenceAssembliesUrl": "https://sgcomcdnponestream.blob.core.windows.net",
  "referenceAssemblyDownloadLocation": "C:\\Users\\{{username}}\\Downloads",
  "discoveryApiTimeoutSeconds": 30,
  "applicationsApiTimeoutSeconds": 30,
  "developerStudioApiTimeoutSeconds": 300,
  "businessRulesApiTimeoutSeconds": 300,
  "assembliesApiTimeoutSeconds": 300,
  "servers": {
    "{{custom_server_url}}": {
      "authorityUrl": "{{custom_ois_url}}",
      "disableSslVerification": true
    }
  }
}
```

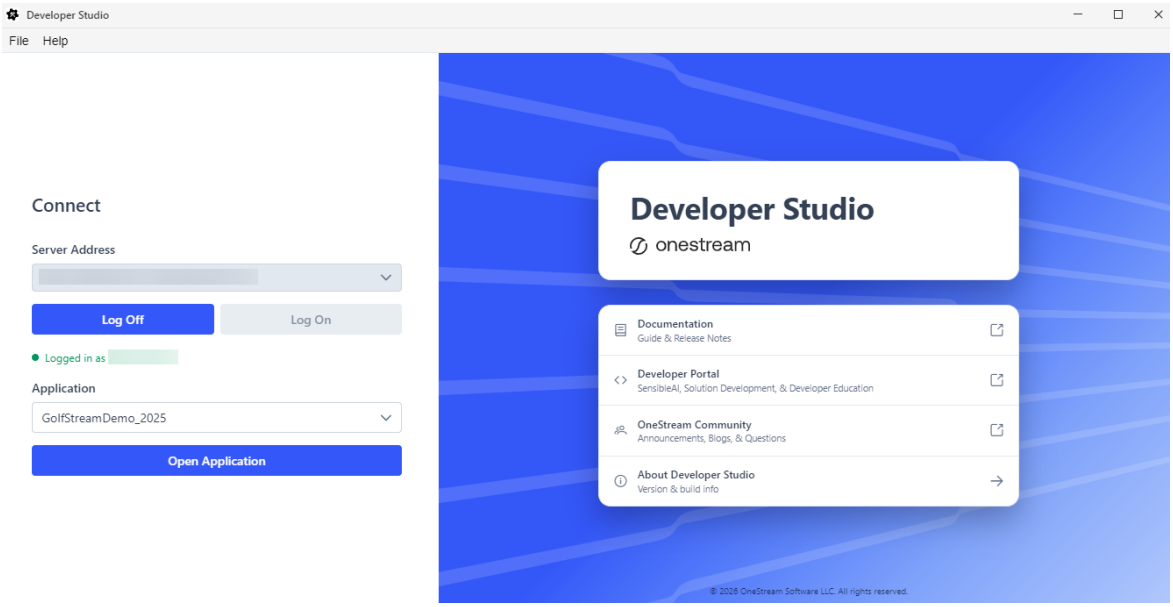
Connect

To connect to a server and use Developer Studio, complete the following steps:

1. On the launch page, in the **Server Address** field, enter a server address. If you have previously connected to a server, you can select the server address from the drop-down menu.

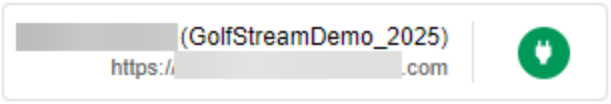
   > **NOTE:** Remove OneStreamWeb from the server address. For example, if your server address is https://my.onestreamserver.com/OneStreamWeb, enter https://my.onestreamserver.com.

2. Select the **Log On** button and complete your OIS logon process.
3. From the **Application** drop-down menu, select an application and then select the **Open Application** button.The most recently opened application on the server is saved and automatically selected in the Application field.



On your first login, you are prompted to create a repository. See Repository Setup.

The home page will display a tile with a green Connection icon to indicate you are connected to a server and application. The tile displays the user, application, and server URL. To access the launch page, you can select the tile or go to File > Connect to a Server.



Complete these steps to switch to a different application:

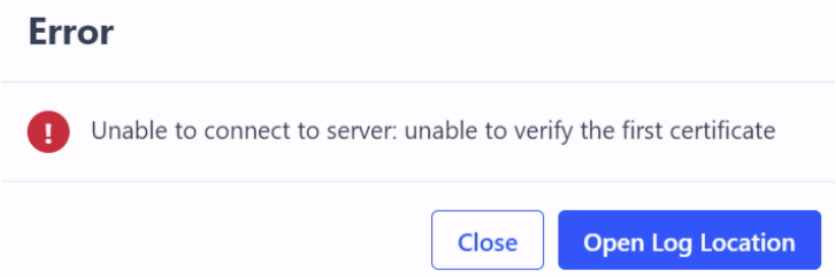
1. Go to **File** > **Connect to a Server**, or select the panel containing the server address and connection icon to open the **Connect** dialog box.

2. From the **Application** drop-down menu, select a different application.
3. Select the **Open Application** button.

If an exception occurs on the server, you will receive an error. Select the **Open Log Location** button to review logging information about the exception.

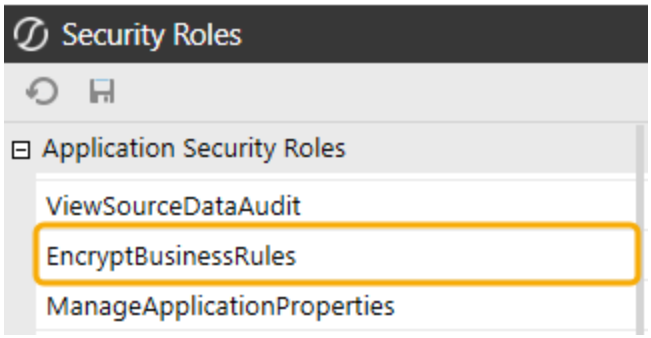
> **NOTE:** Log files are located at AppData\Local\OneStream\Developer Studio\logs.

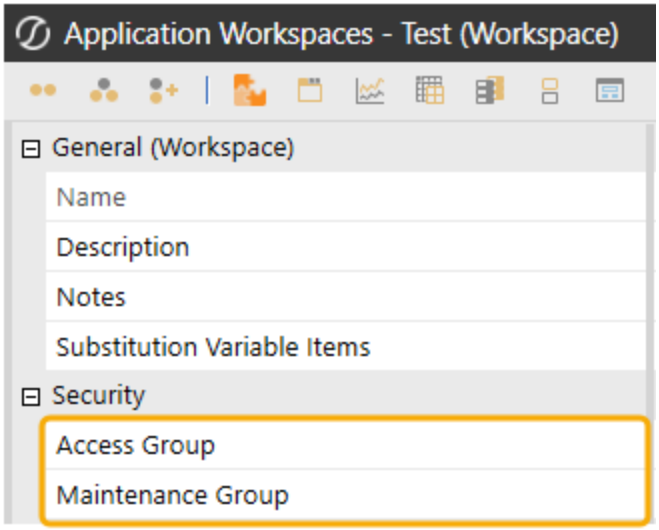
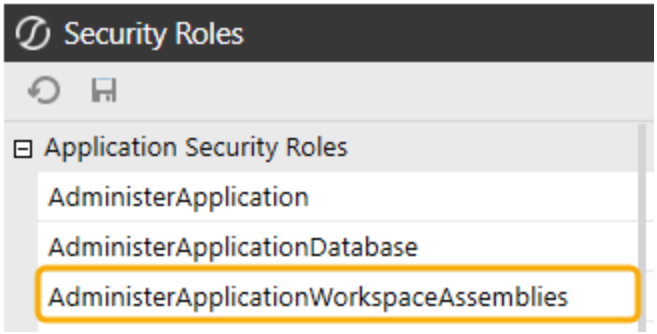
You may need to complete custom environment configuration if you receive an error similar to the following: Unable to connect to server: unable to verify the first certificate. See Setup and Installation.



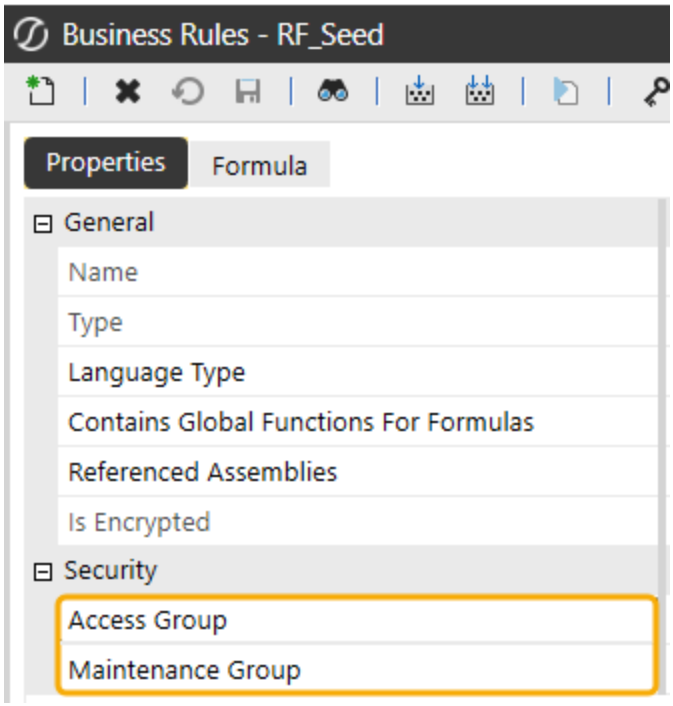
Security Roles

These security roles are necessary to complete Developer Studio functions. Set these roles to groups that will include all Developer Studio users.

Security Role	Developer Studio Function
ApplicationLoadExtractPage	Load/Extract XF Projects
EncryptBusinessRules	Encrypt/Decrypt Business Rules and Workspace Assemblies
	
Workspace Assemblies	
AdministerApplicationWorkspaceAssemblies, Access Group, and Maintenance Group	Push, Pull, and Compile

Security Role	Developer Studio Function
	
	
NOTE: For Workspace assemblies, the Access Group and Maintenance Group selections can be set at both the Workspace and maintenance unit levels on the Workspaces page.	
Business Rules	

Security Role	Developer Studio Function
Access Group and Maintenance Group	Push, Pull, and Compile

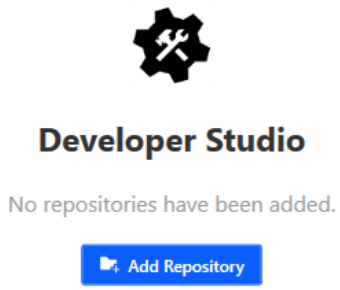


Repository Setup

A repository is a folder location on your local computer. Within that folder location, you will set up one or multiple projects to sync. In a typical setup, a repository represents the projects (business rules, Workspace assemblies, and XF Projects) that belong in a single solution. However, you can choose to use one repository for many solution projects.

To get started in Developer Studio, complete the following steps to create a repository:

1. Select the **Add Repository** button or go to **File** > **Add Repository**.



2. In the **New Repository** dialog box, select the **Browse** icon and use **File Explorer** to select a repository root folder.

> **TIP:** It is recommended that you select an empty folder the first time running Developer Studio. To prevent slow loading times, avoid selecting your user profile folder or any other high-level directory.

3. Enter a name for the repository.

4. Select the **Save** button.

NOTE: If the repository folder selected contains osmap files or .xfproj files, those items automatically display in the Repository Items grid.

Repositories

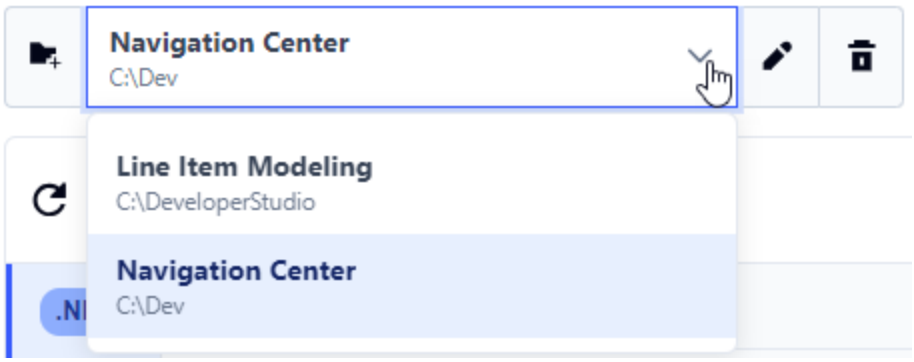
The home page displays once a repository is made. On the home page, you can manage repositories and cycle between them.

To add repositories, select the **Add Repository** icon. You can also go to **File** > **Add Repository**. Select the root folder and name the repository. The repository path you choose is where your project files will be added to. When you add a new repository, the home page defaults to that repository.

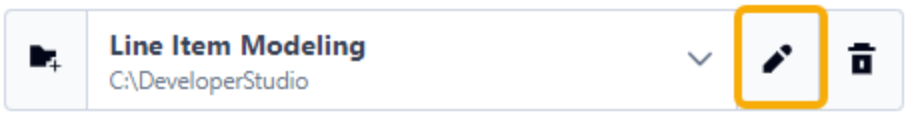
> **NOTE:** The repository name must be unique across your local machine.

To set a default repository root folder, go to **File** > **Settings**. The folder specified here will display as the default repository root folder when creating a new repository. See Settings.

All saved repositories display for selection in the drop-down menu.



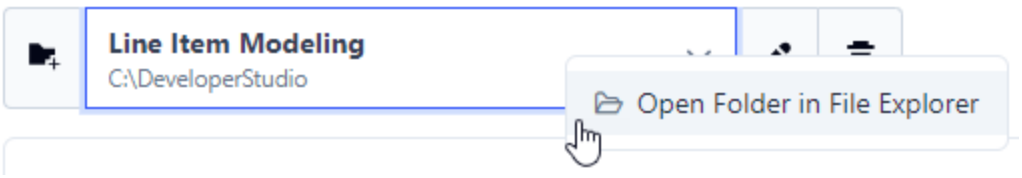
To edit the repository, select it from the drop-down menu and select the **Manage Repository** icon. In the **Manage Repository** dialog box, you can change the folder location and rename the repository.



> **NOTE:** Changing the folder location does not move your items to that folder. It changes which files the repository points to.

To remove a repository, select the **Remove Repository** icon. Removing a repository does not delete content from your local or within OneStream. Only the repository reference is deleted.

To view the folder location of the selected repository, right-click on the repository object and select **Open Folder in File Explorer**.



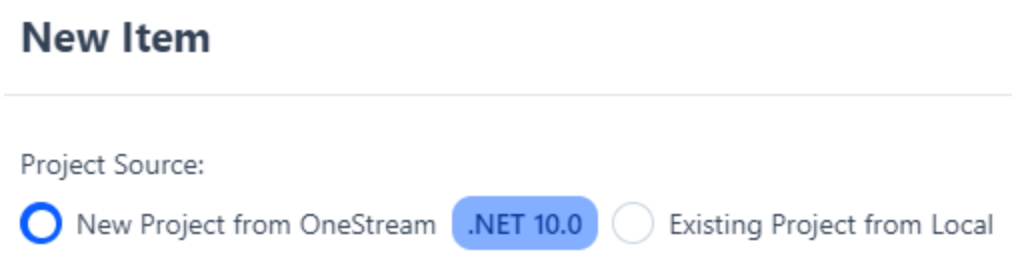
If a repository folder has been altered or deleted, the repository item will display an error icon in the drop-down menu to indicate that the repository location does not exist.

.NET Items

A .NET item is a business rule or Workspace assembly from OneStream that you can pull to your local. Once you have added an item, you can add, edit, delete, and rename files and folders from the IDE of your choice and sync those changes with the server.

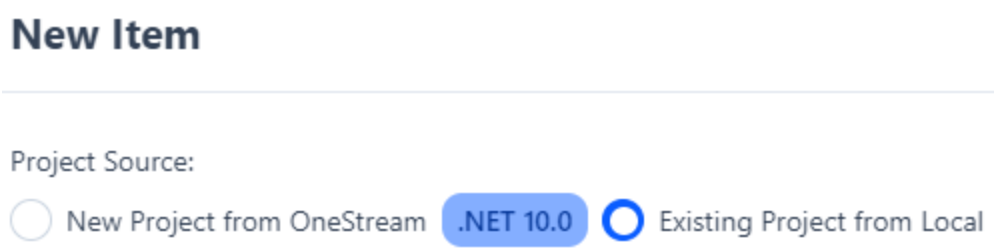
There are two ways you can create items:

- If you do not have a .NET project defined locally and want to create one based on what you have in OneStream, your Project Source selection should be New Project from OneStream. This selection creates a .NET project file, .csproj or .vbproj, and pulls down the selected OneStream files. See New Project From OneStream.



- If you have an existing project locally and want to map it to a business rule or Workspace assembly from OneStream to manage, your Project Source selection should be Existing

Project from Local. See Existing Project From Local.

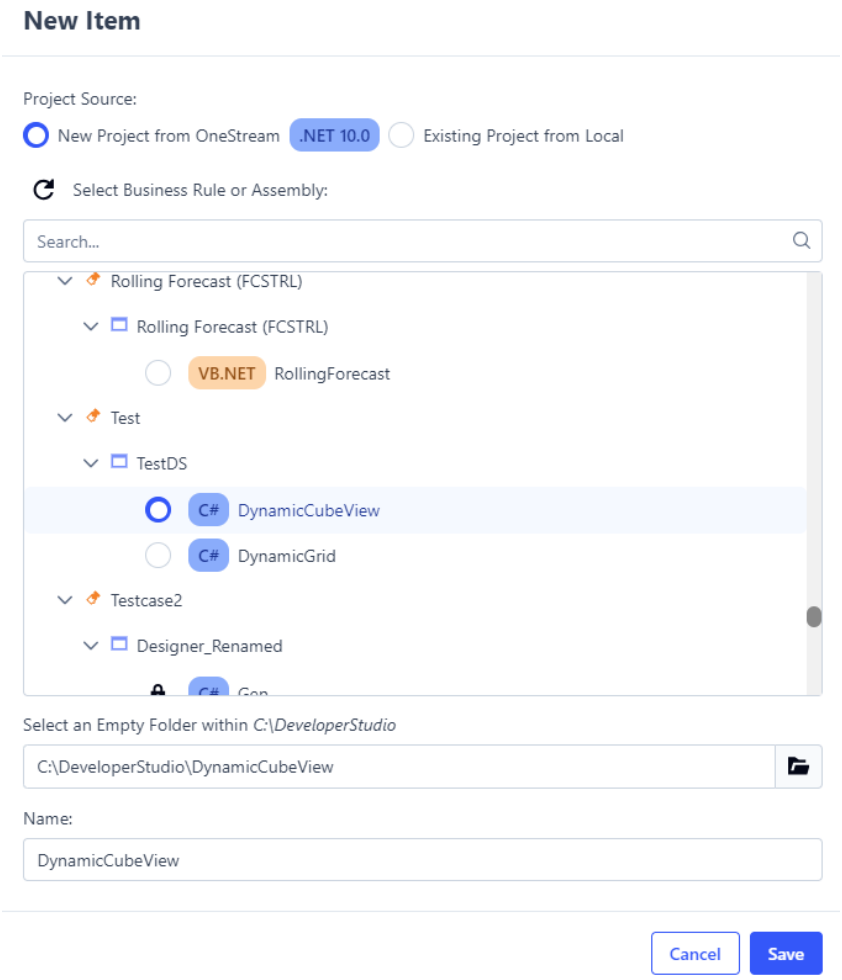


New Project From OneStream

Complete these steps to add an item and create a new project from OneStream.

1. Select the **Add Item** button to display the New Item window.
2. Under **Project Source**, select **New Project from OneStream**.

   In the navigation tree, select a business rule or assembly. The project folder you will select maps to the business rule or assembly selected. The navigation tree displays the compiler language for each item.

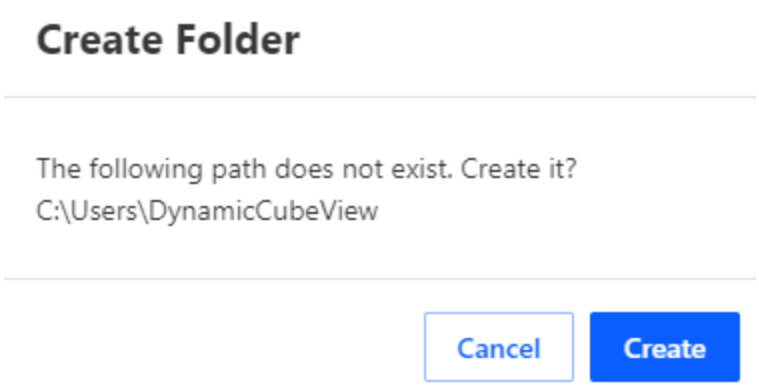


To refresh business rules and assemblies within the navigation tree, select the **Refresh** button.

Items with this icon are encrypted and cannot be selected until they are decrypted. See Encrypt and Decrypt Items.

3. The Select an Empty Folder field defaults to the repository root path. After selecting a business rule or assembly, the field updates automatically to include a subfolder with the item name. To change the folder, select the **Browse** icon and use **File Explorer** to select an empty folder in that repository.

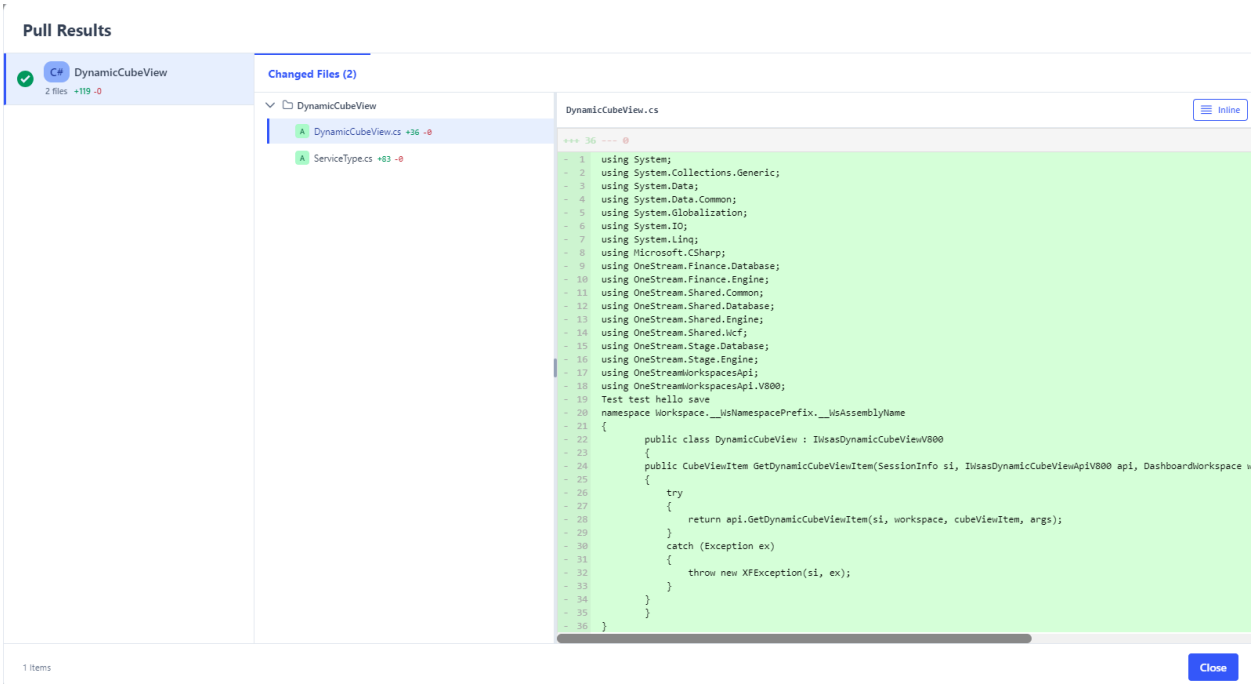
   If the selected folder does not exist, the Create Folder dialog box displays when you select the Save button. Select the **Create** button to create a folder with the name of the assembly or business rule selected.



> **NOTE:** You will receive an error if the folder you selected is not empty or not within the repository directory.

4. The Name field defaults to the selected assembly or business rule name. Rename the item if needed. The item name is local to Developer Studio and does not affect the OneStream server in any way.
5. Select the **Save** button.

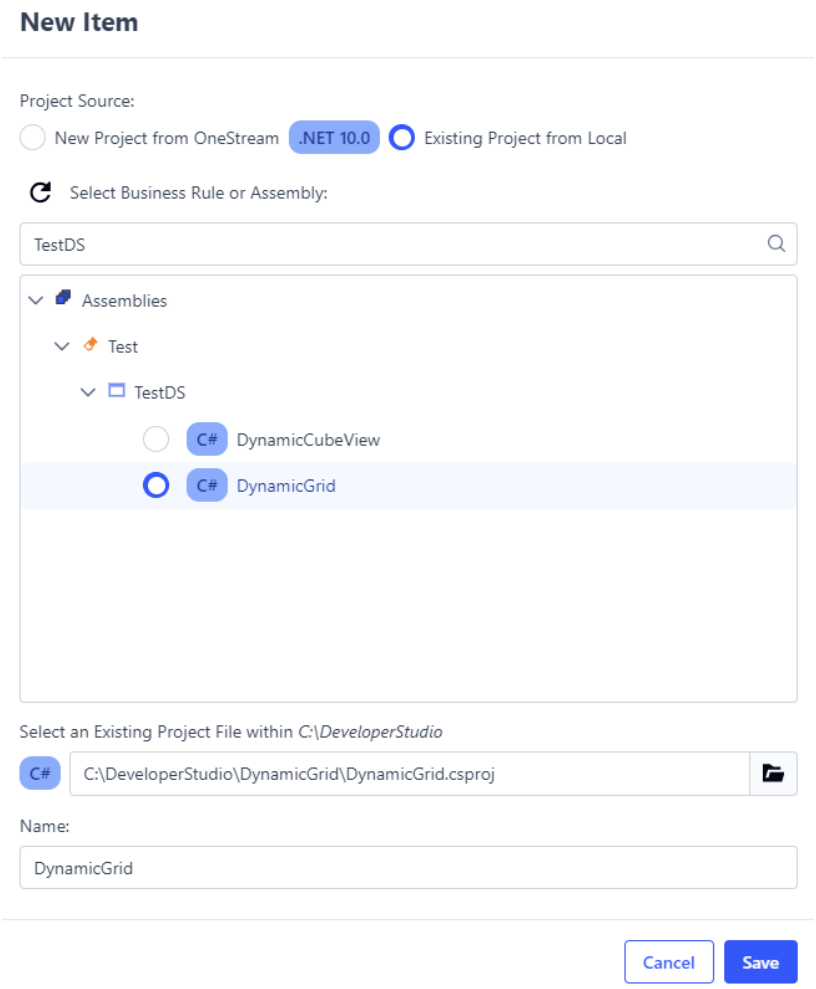
The Pull Results window will display the item and all files that were pulled down from OneStream to your local. See Push and Pull .



Existing Project From Local

Complete these steps to add an item to an existing project from local:

1. Select the **Add Item** button to display the **New Item** window.
2. Under **Project Source**, select **Existing Project from Local**.
3. In the navigation tree, select a business rule or assembly. The navigation tree only displays items with the same compiler language as the project file you select. All local files within the project file you select will map to the business rule or assembly selected from the navigation tree.



To refresh business rules and assemblies, select the **Refresh** icon.

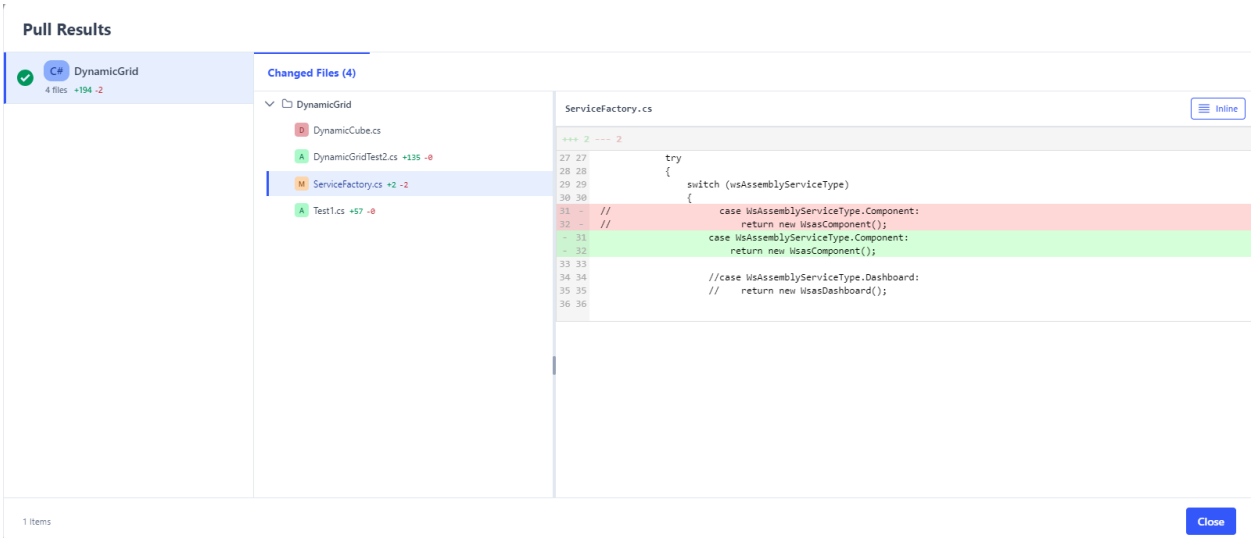
Items with this icon are encrypted and cannot be selected until they are decrypted. See Encrypt and Decrypt Items.

4. Select the **Browse** icon and use **File Explorer** to select an existing project file within your repository. The compiler language displays next to the file path after selecting a project file.

   > **NOTE:** The project file must exist within the repository directory.

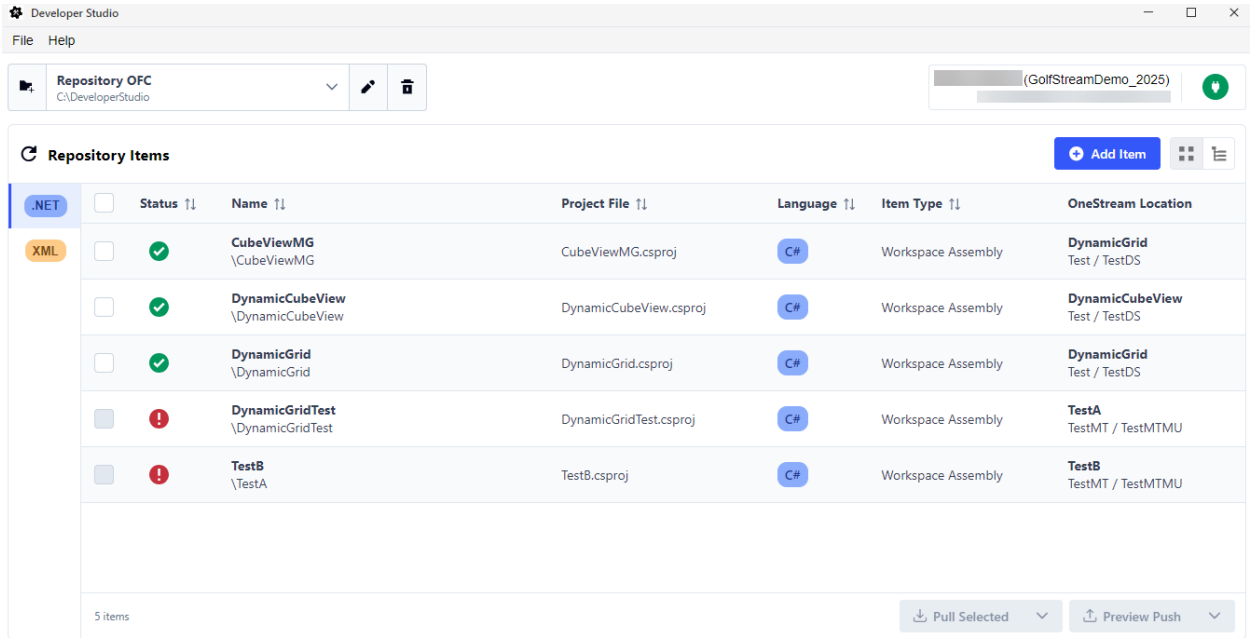
5. The Name field defaults to the project file name. Rename the item if needed. The item name is local to Developer Studio and does not affect the OneStream server in any way.
6. Select the **Save** button.
7. The Pull Latest Code dialog box displays when you save the item.
   a. To overwrite your local project files with the chosen business rule or assembly files, select the **Pull Latest** button.
   b. To keep what you have locally in your project file, select the **Do Not Pull** button.

If you selected Pull Latest, the Pull Results window displays showing the item that was pulled to your local and all files created, updated, or deleted. See Push and Pull .



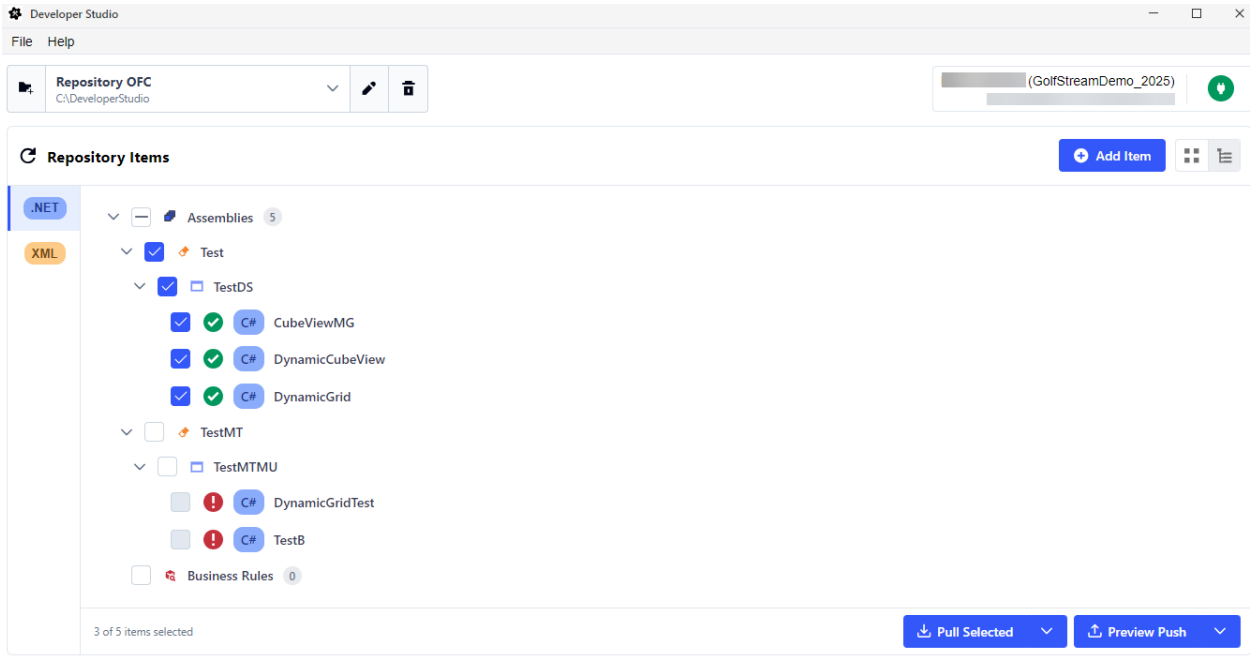
.NET Repository Items Grid

All .NET items created display in the Repository Items grid.



To view items in a hierarchy view, select the **Hierarchy View** icon. This view enables you to easily select all business rule items and select all assemblies under a specific Workspace or maintenance unit. Select the **Grid View** icon to switch back to a grid view.

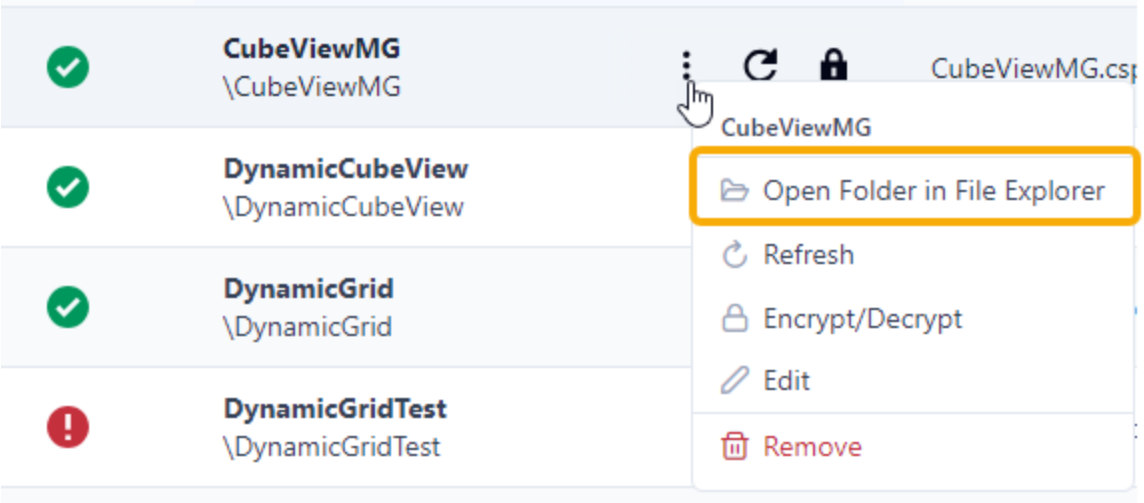
.NET Items



All items within the currently selected repository display in the Repository Items grid with the following columns:

Column	Description
Status	The status of the item:  **Ready**: The item is valid and ready to push and pull.  **Conflicts**: The item has conflicts with the server that must be resolved before you can push and pull. For example, if the item you are trying to push has been encrypted on the server.  **Error**: The item has errors that must be resolved before you can push and pull. For example, if the business rule or workspace assembly does not exist on the server.
Name	The name of the item and the full file path.
Project File	The name of the project file.
Language	The languages of the project file, which can be C# or VB.NET.
Item Type	The item type is either business rule or workspace assembly.
OneStream Location	The location of the item within OneStream. For example, WorkspaceA - MUA - Assembly A. WorkspaceA is the Workspace name, MUA is the Maintenance Unit name, and Assembly A is the assembly name.

To view the folder file location of an item, right-click the item or select the ellipses to display the context menu, then select **Open Folder in File Explorer**.

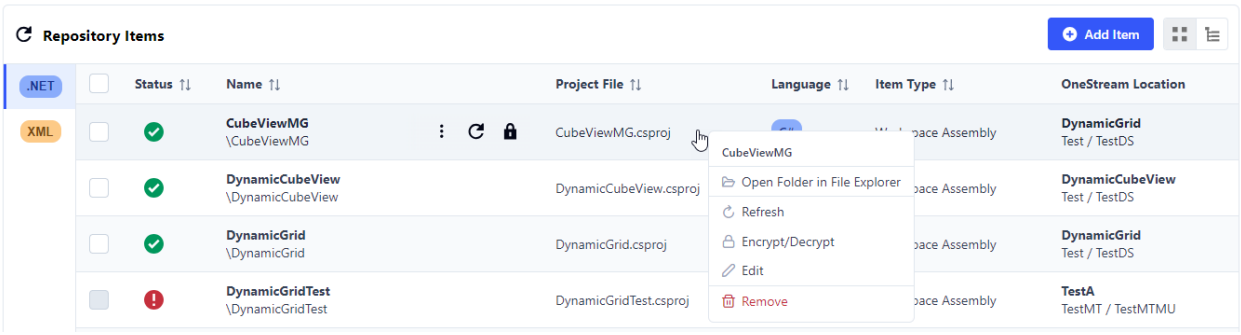


Repository Items

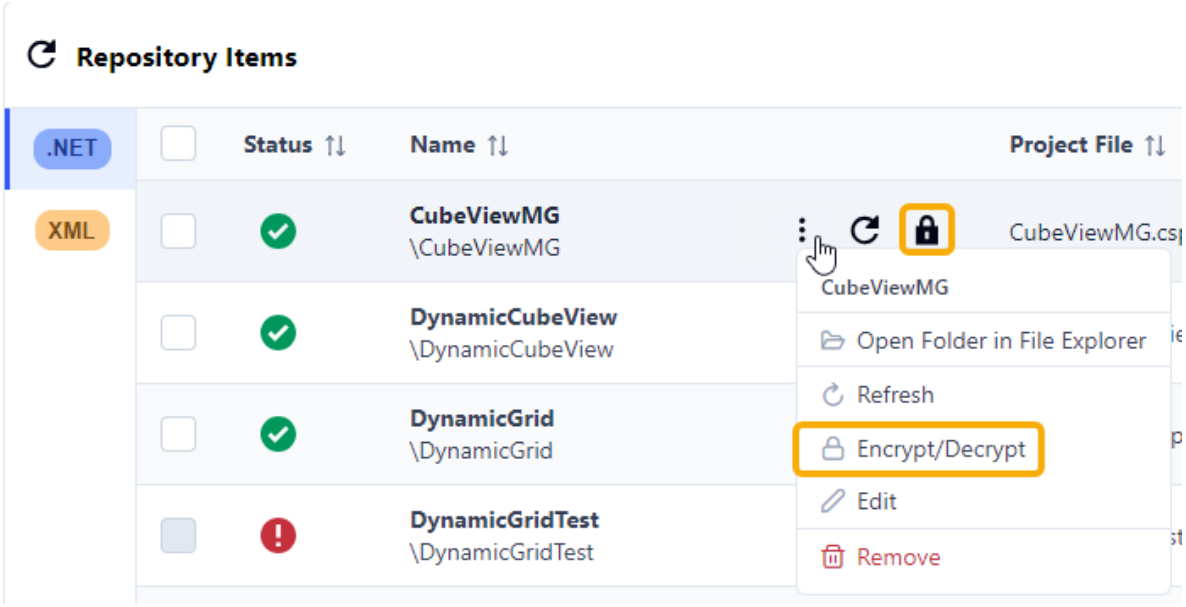
To manually refresh the Repository Items grid, select the **Refresh** icon.

> **NOTE:** If you edit your osmap files outside of Developer Studio, the grid will automatically refresh.

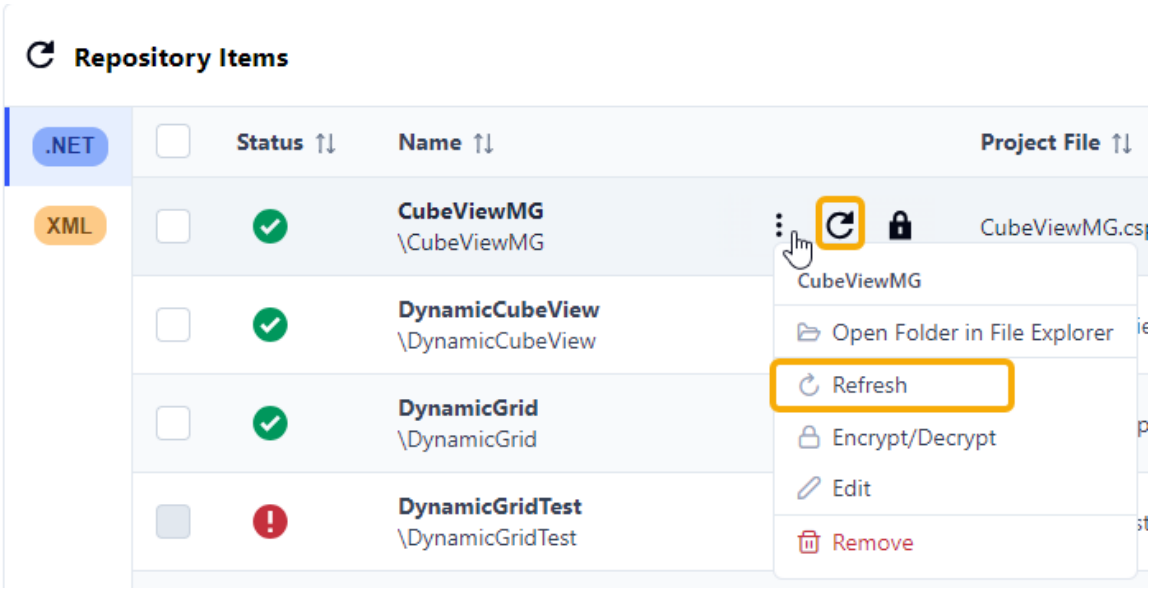
To manage repository items, hover over the item or right-click to display available actions.



To encrypt or decrypt an item, select the **Encrypt/Decrypt** icon or choose the option from the context menu. See Encrypt and Decrypt Items.



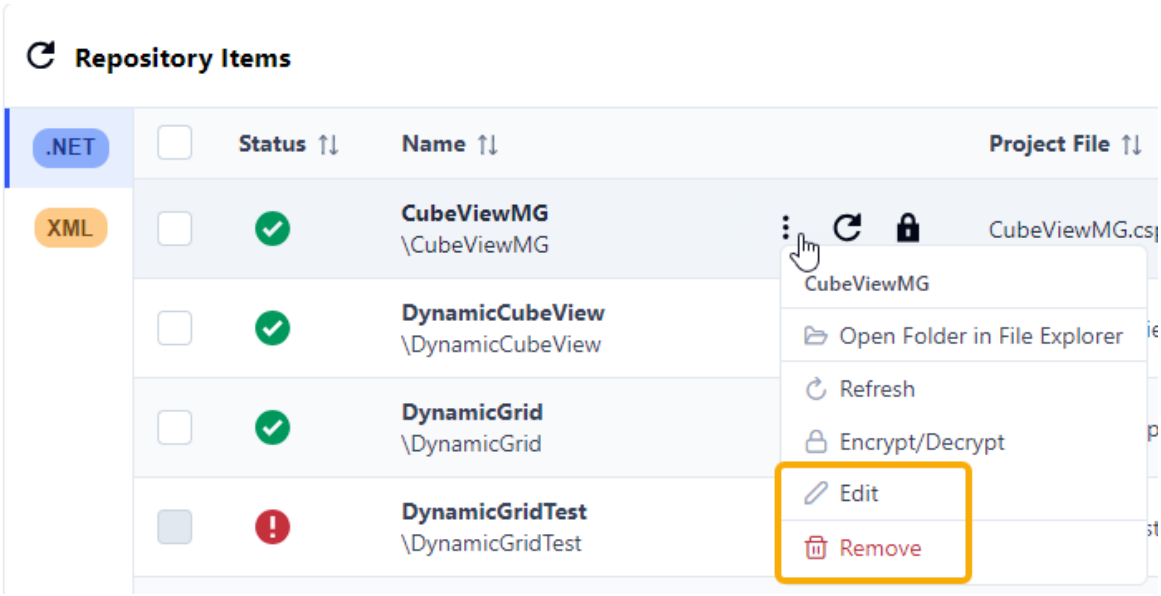
To manually refresh a single item, select the **Refresh** icon or choose the option from the context menu.



To edit or remove an item, select the ellipses to display the context menu and select an option.

When you select **Edit**, the Edit Item window displays. You can choose a new business rule or assembly from OneStream. You cannot rename an item or edit the selected project file.

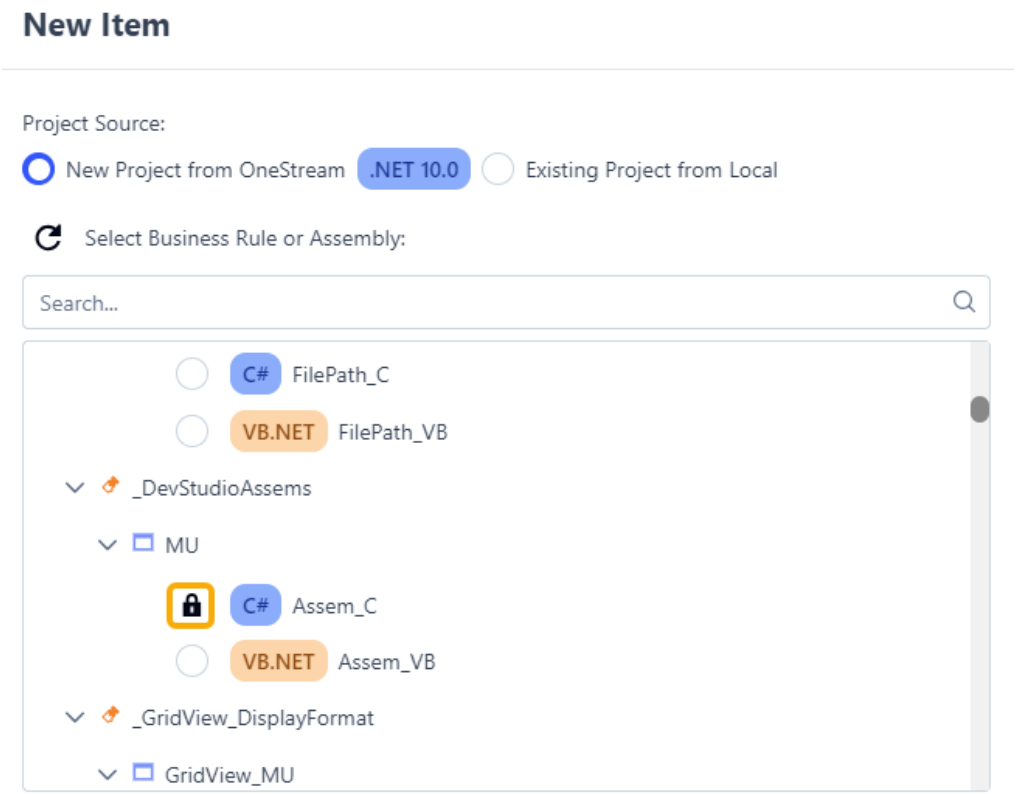
> **IMPORTANT:** When you remove an item, nothing is deleted locally or on the server. If you want to delete your project or source file, this must be done manually outside of Developer Studio. If the same item is created in two different repositories with different folder paths, removing the item from one repository will not remove it from the other.



Encrypt and Decrypt Items

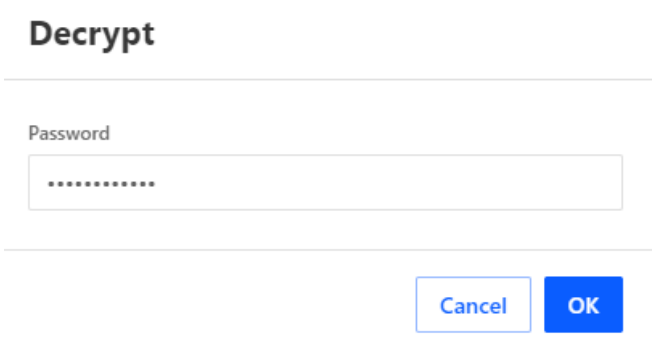
Before you can work with an item in Developer Studio, all its contents must be available unencrypted on the server. To decrypt a file before pulling it to your local, complete these steps:

1. In the **New Item** or **Edit Item** window, select the **Decrypt** icon for an encrypted item.

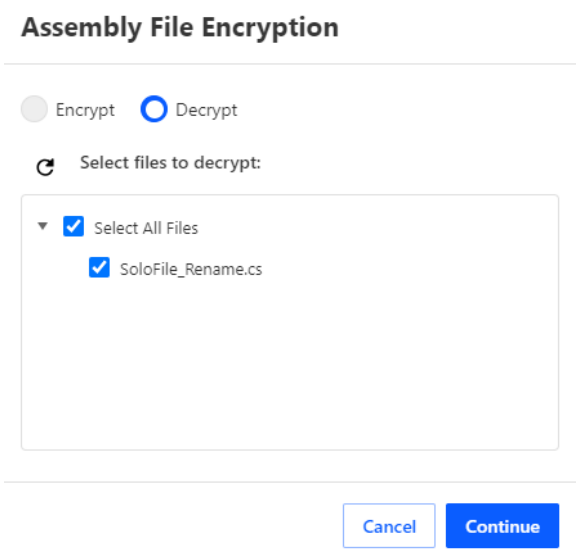


2. The window that displays depends on whether you are decrypting a business rule or a workspace assembly file.

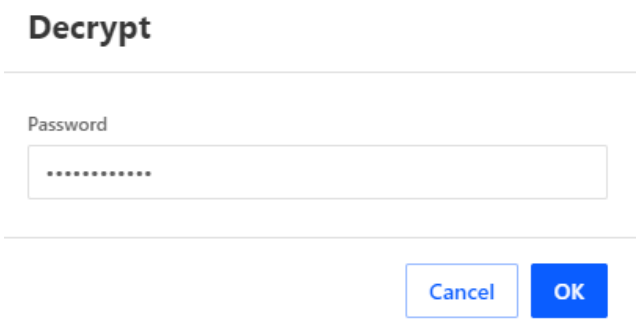
a. If you are decrypting a business rule, the Decrypt dialog box displays. Enter a valid password and select the **OK** button.



b. If you are decrypting an assembly file, the Assembly File Encryption dialog box displays. Use the navigation tree to select files to decrypt and select the **Continue** button.

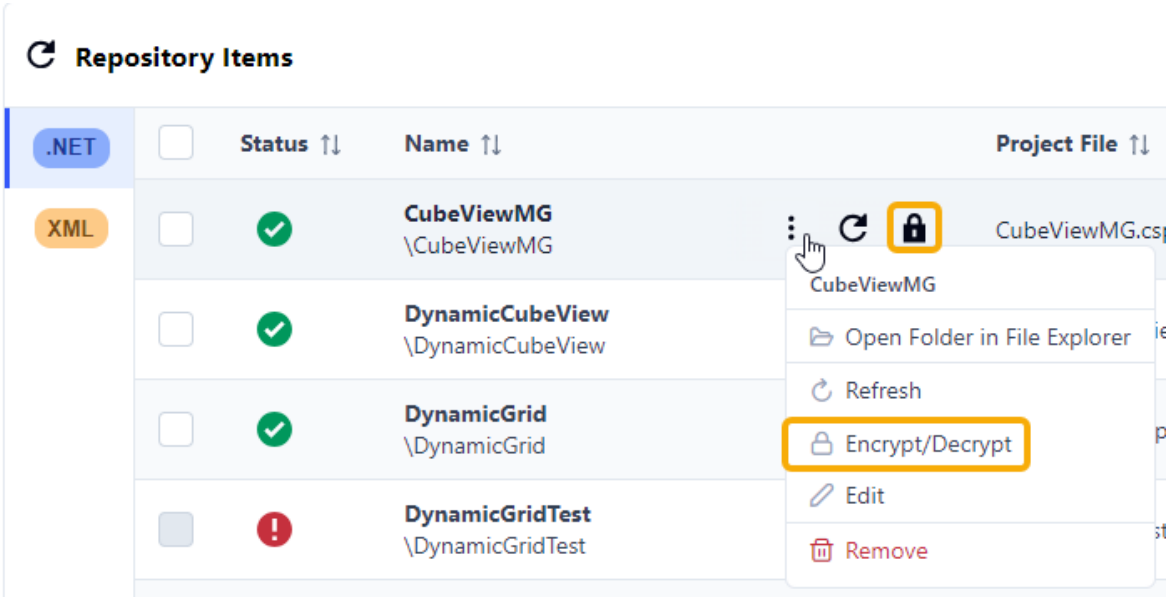


In the **Decrypt** dialog box, enter a valid password and select the **OK** button.



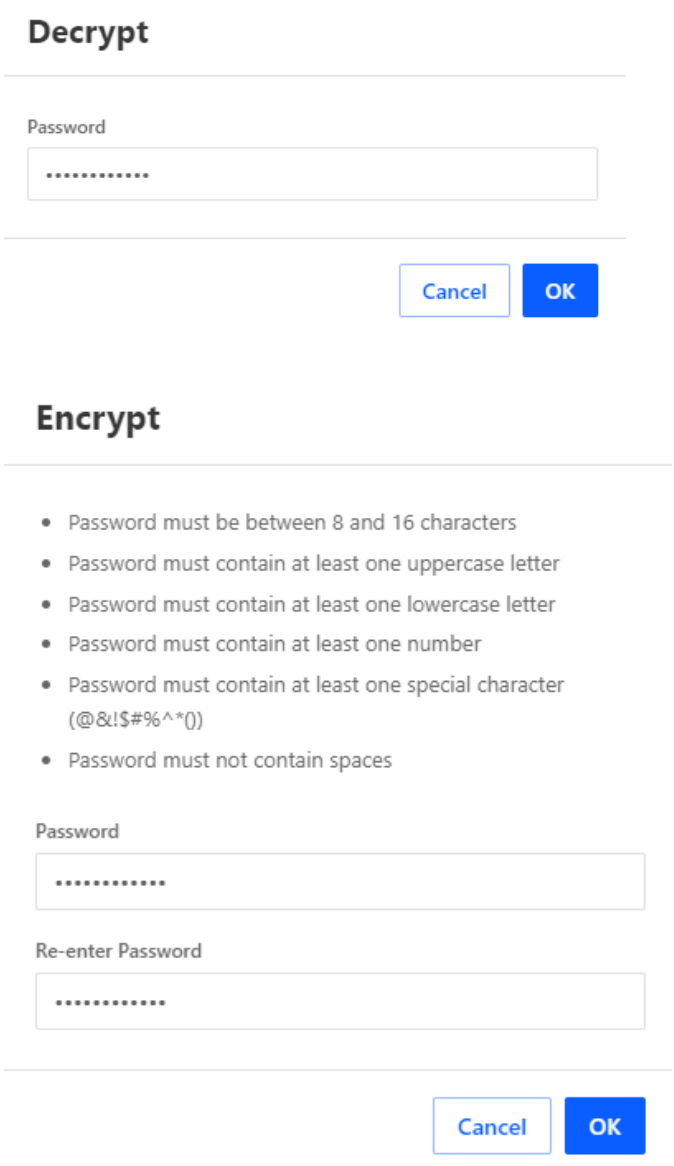
To encrypt or decrypt an item in the Repository Grid, complete these steps:

1. Select the **Encrypt/Decrypt** option for that item.



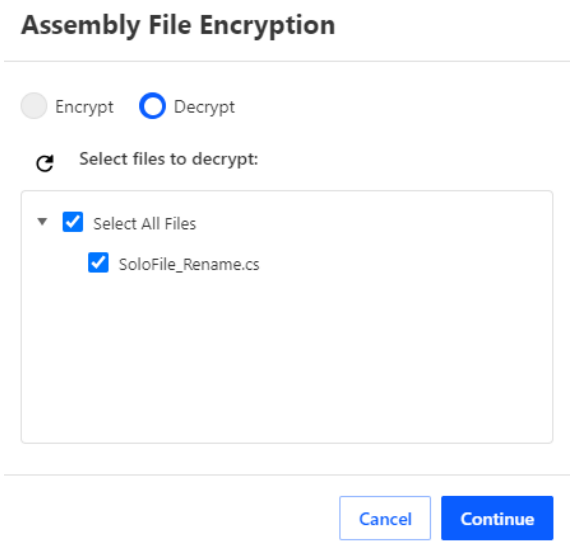
2. The window that displays depends on whether you are encrypting/decrypting a business rule or assembly file.

a. If you are encrypting or decrypting a business rule, the Decrypt or Encrypt dialog box displays. Enter or create a valid password and select the **OK** button.

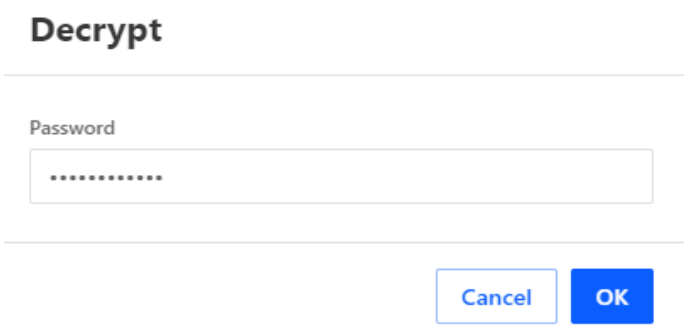


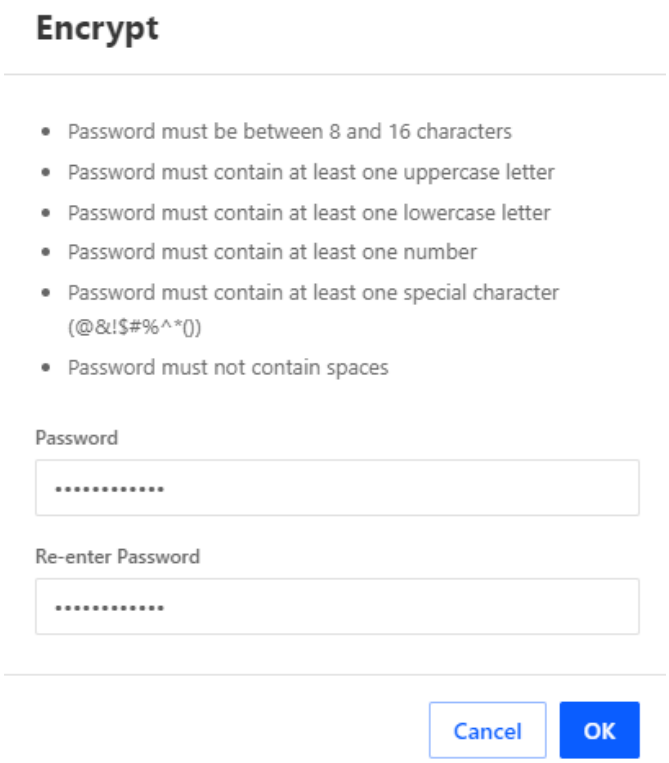
b. If you are encrypting or decrypting an assembly file, the Assembly File Encryption dialog box displays. Select either **Encrypt** or **Decrypt** and then use the navigation tree to select files. Select the **Continue** button.

> **NOTE:** Once a single file in an assembly is encrypted, you cannot synchronize any file in the assembly with Developer Studio.



Enter or create a valid password and select the **OK** button.




Encrypt
Password must be between 8 and 16 characters
Password must contain at least one uppercase letter
Password must contain at least one lowercase letter
Password must contain at least one number
Password must contain at least one special character (@&!$#%^*())
Password must not contain spaces
Password
Re-enter Password
Cancel
OK

Push and Pull

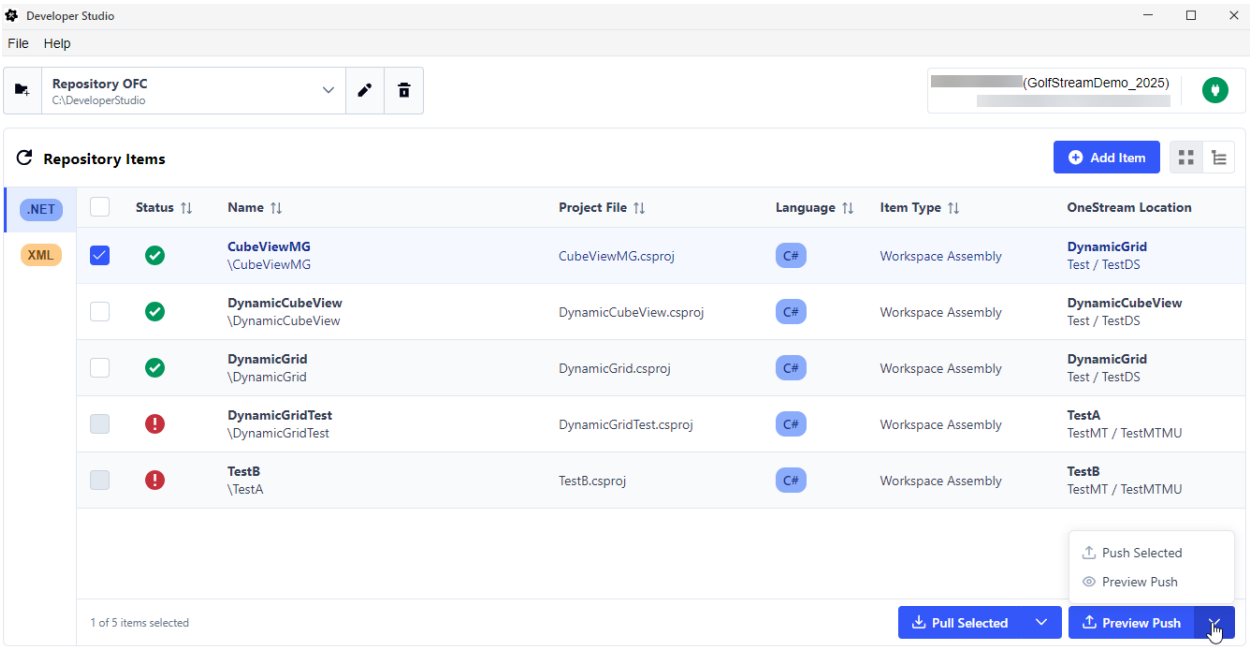
Pushing and pulling enables you to keep your local files and your OneStream files in sync. Performing a push replaces the items on the OneStream server with your local content. Performing a pull replaces items on your local machine with OneStream server content.

Pushing and pulling only impacts compilable files, such as C# or VB.NET files. If you have a non-compilable file in your local project, pulling from the server will not impact that file.

> **NOTE:** Items with the status Conflicts or Errors cannot be pushed or pulled.

Before pushing or pulling, you can view a preview of the results and review changes, providing easy visibility into incoming and outgoing changes before you apply them. After an item is pushed or pulled, a results window displays a confirmation of the changes.

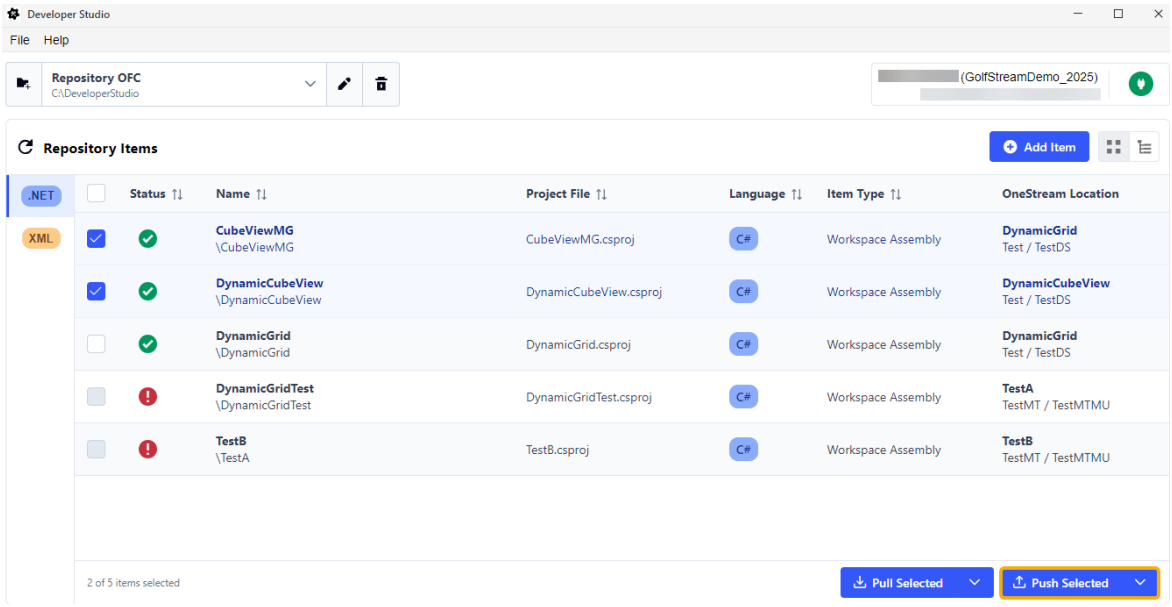
When you select the drop-down arrow for a push or pull operation, the Push/Pull Selected and Preview Push/Pull options display. The option you choose becomes the default for following push/pull operations until a different value is selected.



Push

To push your changes to OneStream, complete these steps:

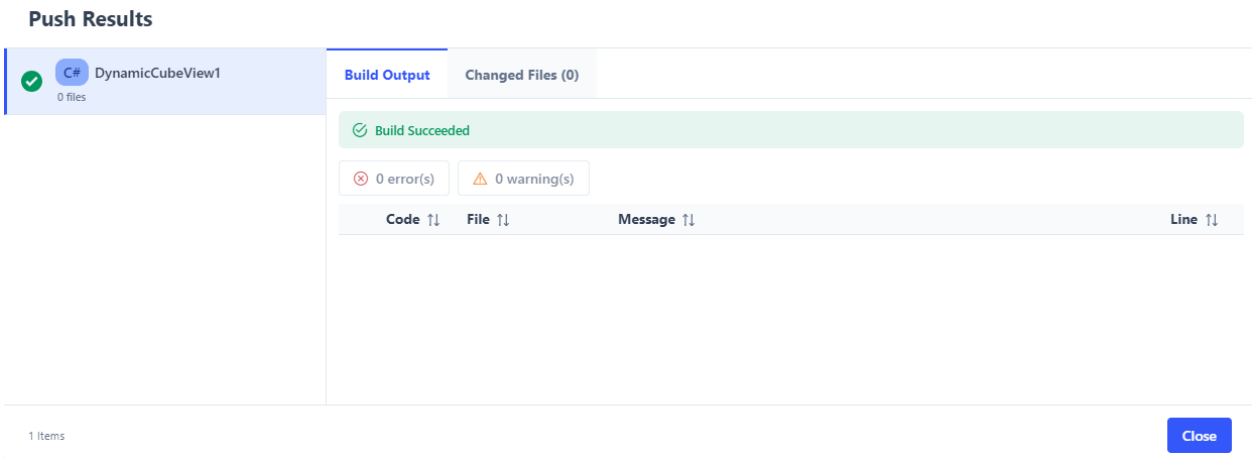
1. Complete and save the changes in your local files.
2. In the **Repository Items** grid, use the checkboxes to select the items you want to push. To select all items in the repository, select the multiselect checkbox.
3. Select your default push operation.



4. The Confirm Push Operation dialog box displays. To confirm that you want to replace items on the OneStream server with your local content, select the **OK** button.

   To prevent the Confirm Push Operation dialog box from displaying again, select the **Don't show this confirmation again for push and pull operations** checkbox.

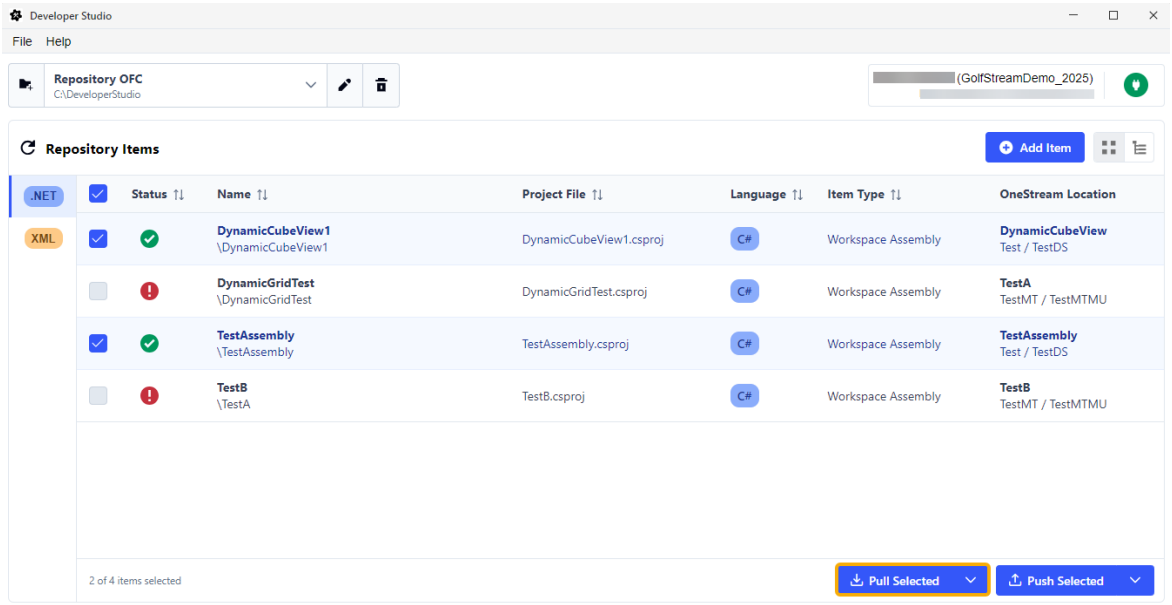
Once a push is completed, the Push Results window displays showing updates to the files. A push compiles on the server, and the Build Output tab displays compilation results



Pull

To pull content from OneStream down to your local, complete these steps:

1. Complete and save your changes in OneStream.
2. Use the checkboxes to select the items you want to pull these changes into. To select all items in the repository, select the multiselect checkbox.
3. Select your default pull operation.

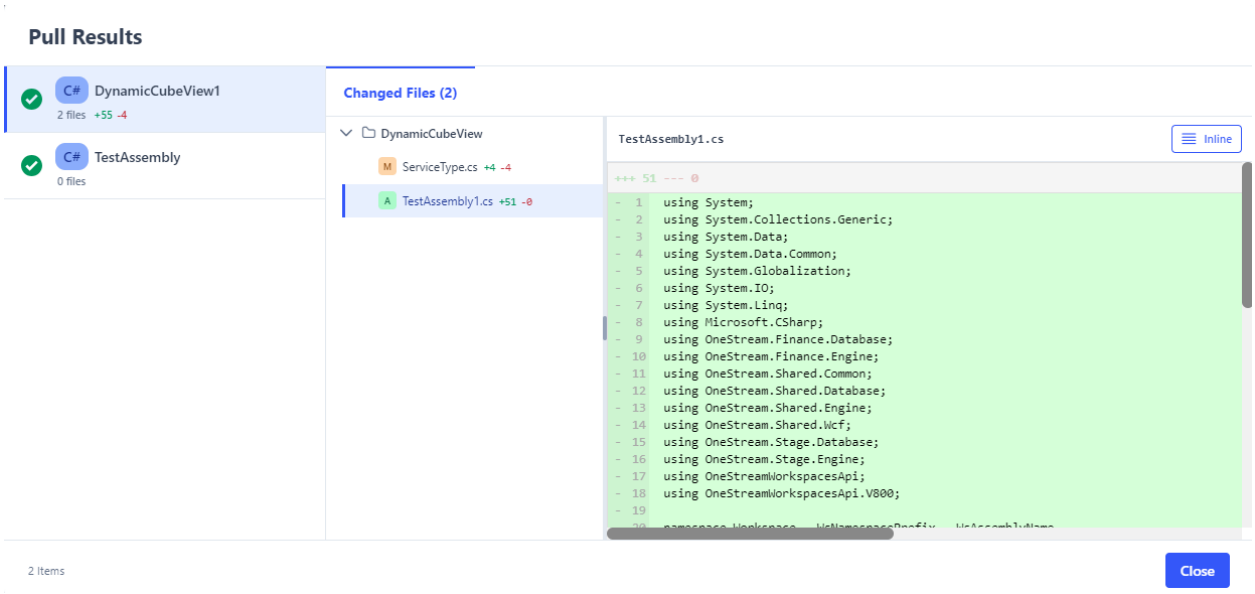


4. The Confirm Pull Operation dialog box displays. To confirm that you want to replace items on your local machine with OneStream server content, click the **OK** button.

   To prevent the Confirm Pull Operation dialog box from displaying again, select the **Don't show this confirmation again for push and pull operations** checkbox.

   > **CAUTION:** If you have added a file locally to any of the selected project items that have not been pushed to the server, performing a pull will delete this local file.

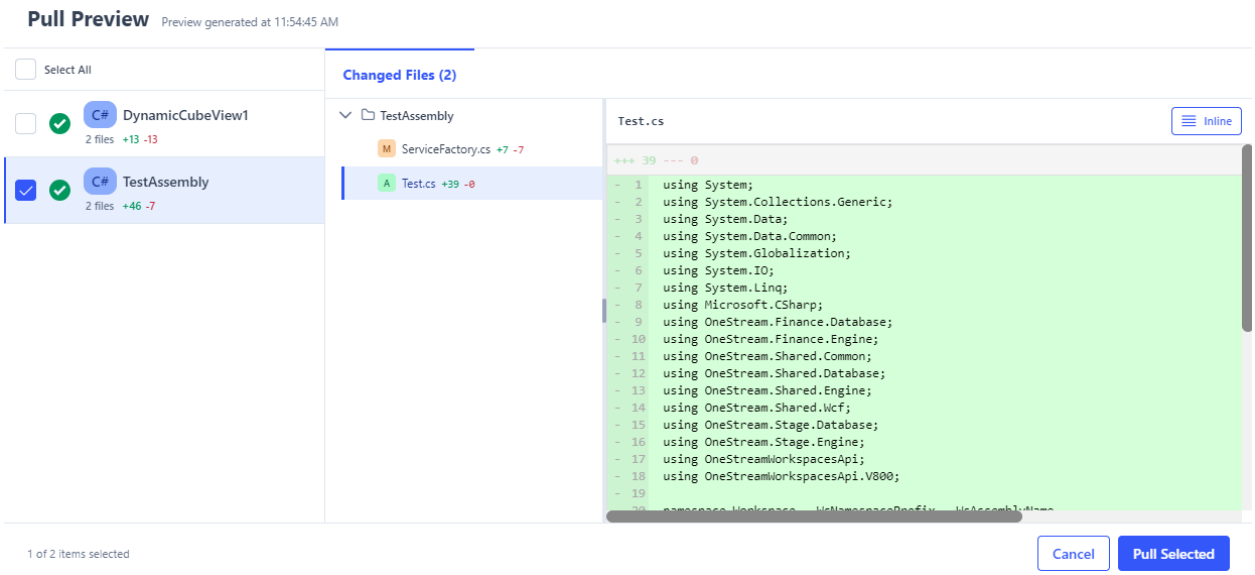
Once a pull is completed, the Pull Results window displays showing updates to your local files.



Previews and Results

The Preview and Results windows display all changed files and provide visibility into line-by-line changes before and after push/pull operations. The left pane displays each item selected for the push/pull operation.

In the Push/Pull Preview window, all items in this pane are selected by default. Use the checkboxes to deselect any items you want to exclude from the push/pull operation.



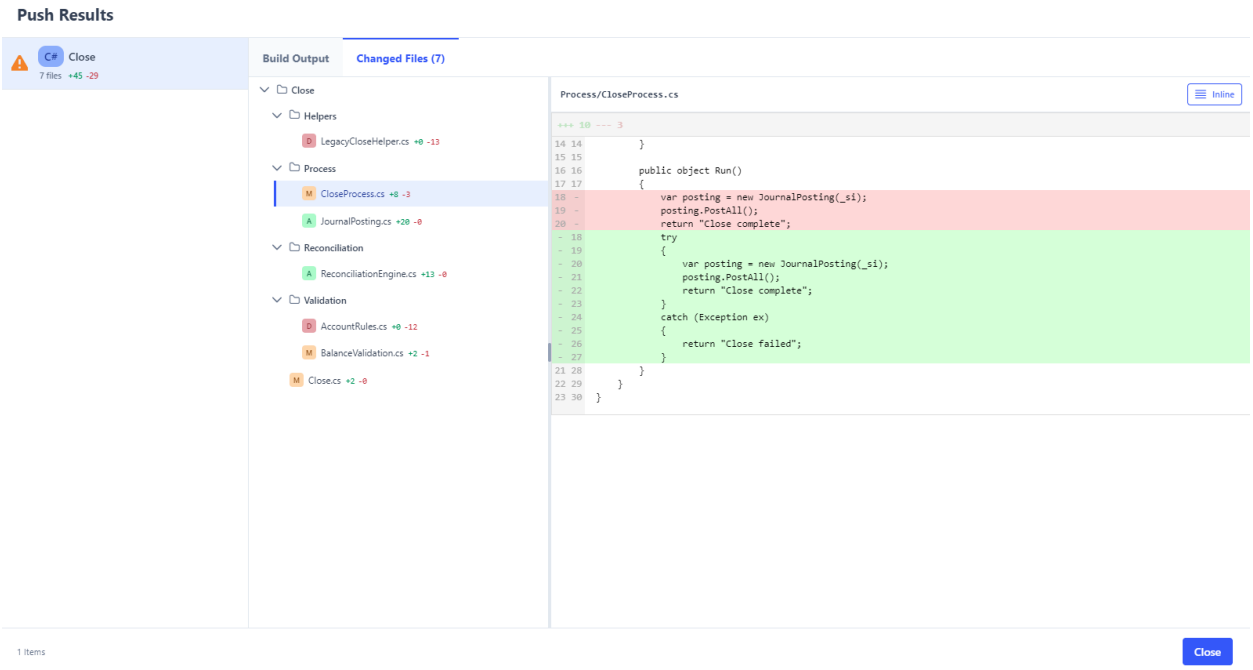
Each item displays with the following indicators:

- Status
  - **Ready**: The item is valid and ready to push and pull.
  - **Conflicts**: The item has conflicts with the server that must be resolved before you can push and pull.
  - **Error**: The item has errors that must be resolved before you can push and pull.
- Item name
- The languages of the project file, which can be C# or VB.NET.
- The total number of files altered
- The total number of lines added and removed

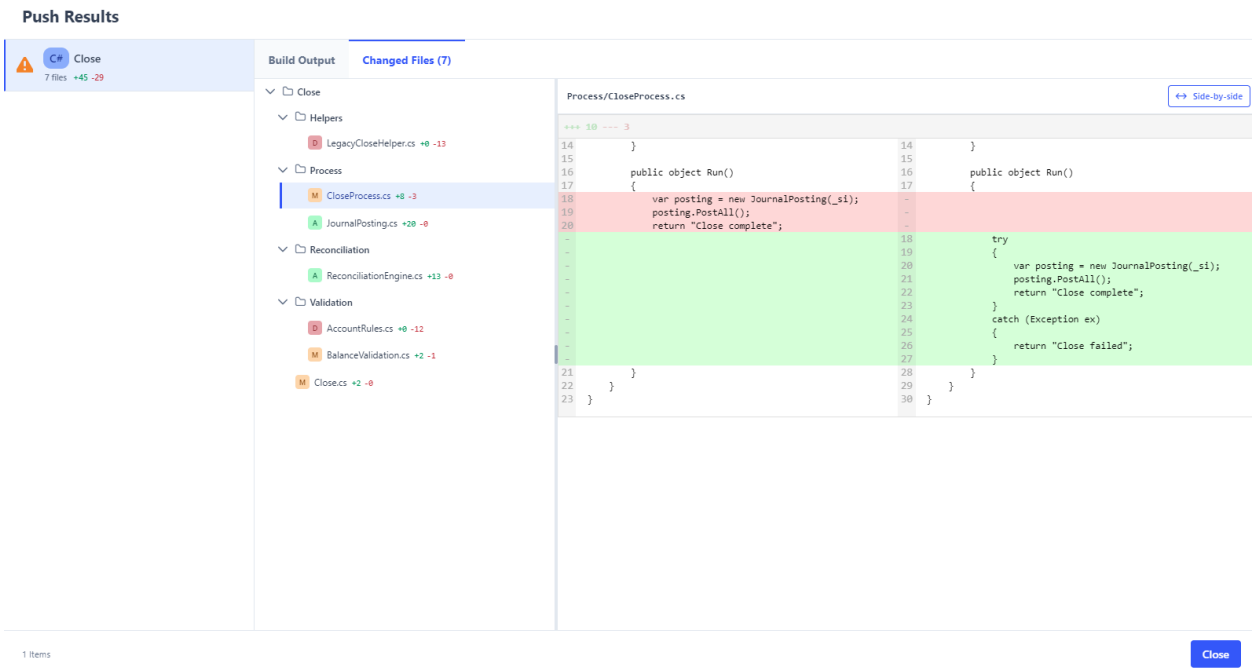
The Changed Files tab displays a tree view of changes for the items selected in the pane. Each file displays the total number of lines changed and a change-type indicator:

- A **Added**: The file was added.
- D **Deleted**: The file was deleted.
- M **Modified**: The file was modified.

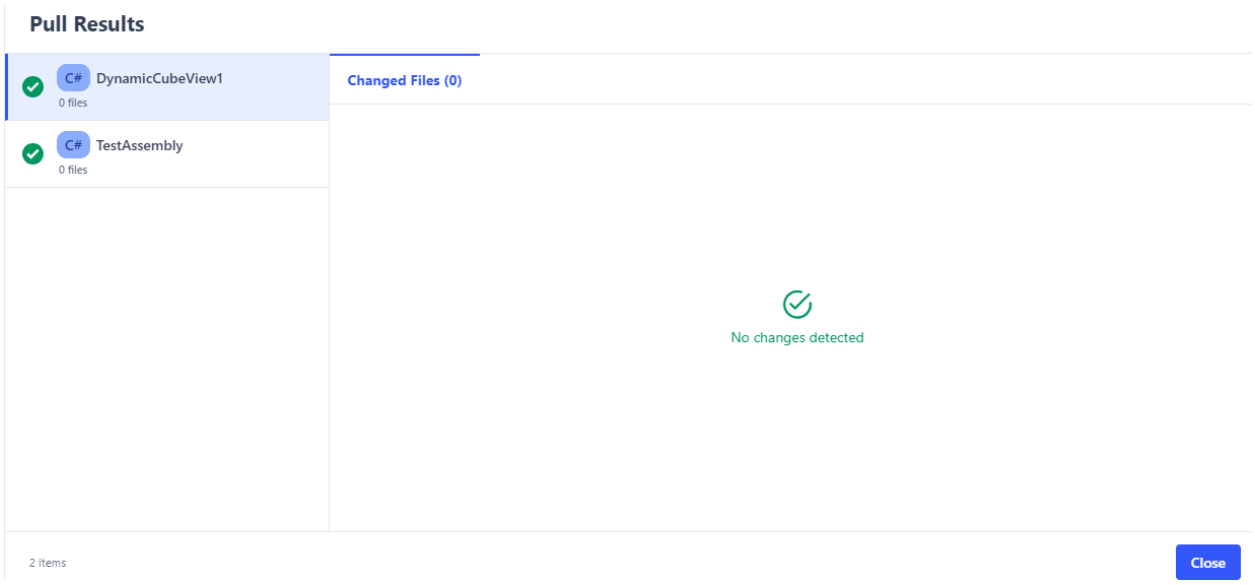
When a file is selected in the navigation tree, the file viewer pane displays the modified lines.



The file viewer defaults to an inline view. To display a side-by-side view, select the **Inline** button.

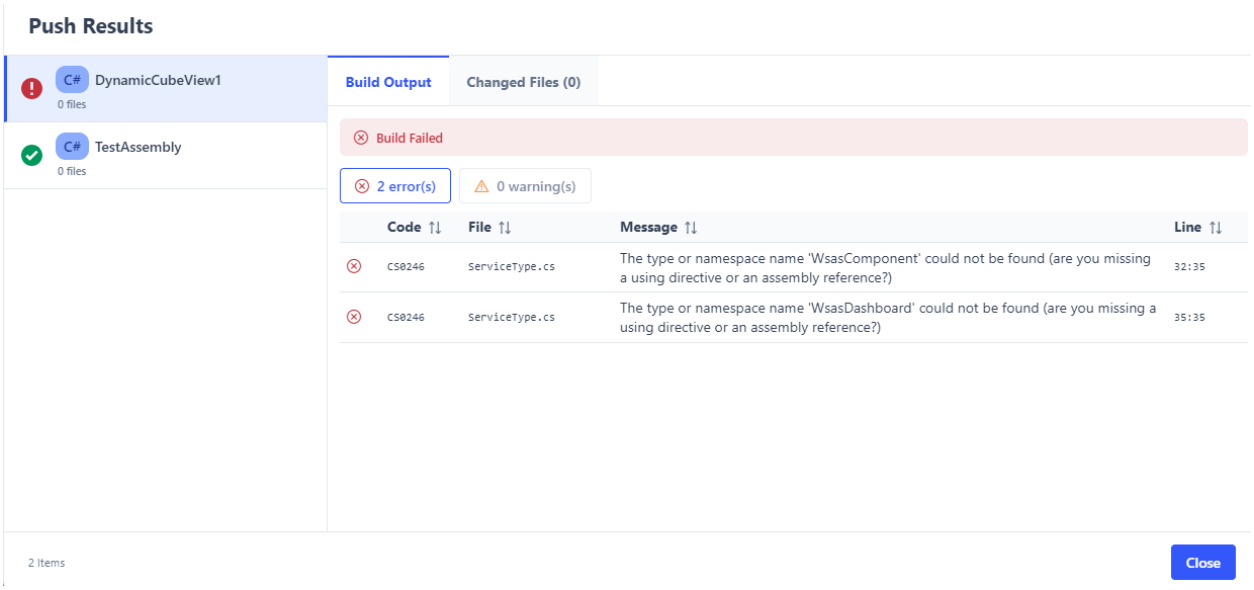


If there are no changes being pushed or pulled, the window displays No changes detected.



Push Results

A push operation compiles on the server, and the Push Results window displays the build output results with any errors or warnings. A banner displays above the grid indicating the compilation status. The grid shows the code for each error or warning, including the associated file and line/column number. You can filter the grid to display only errors or warnings.



XML Items

The XML tab enables you to efficiently manage and package OneStream objects and solutions with XF Projects. See "XF Projects" in the *Design and Reference Guide*. In this tab, you can:

- Create new XF Project files
- Edit and manage existing XF Project files
- Load and extract XF Project files

> **IMPORTANT:** In the XML tab, the available actions and configuration options depend on your current platform version. For example, Platform Version 9.4.0 or later is required to perform load and extract operations. See Appendix B: XML Project Item Configuration and Compatibility.

Manage XML Items

An XML item represents an XF Project file. When you create an XF Project file, Developer Studio enables you to select application items from the OneStream environment you are connected to. The XF Project file template is generated from the OneStream environment items you select during creation. You can also open an existing XF Project file, view the contents, and add or remove items.

Create XF Project Files

To create an XF Project file, complete the following steps:

1. Select the **Add Item** button.
2. In the New XF Project window, configure the following properties:

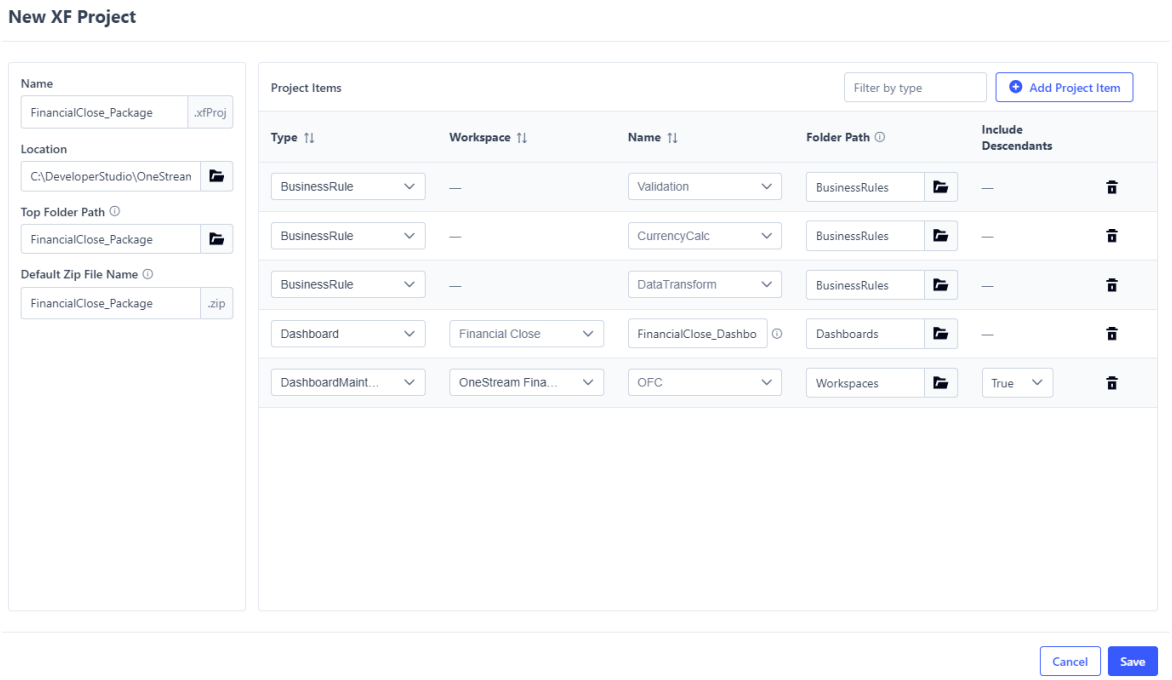
Property	Description
Name	The name must be unique. When you enter a name, .xfproj is automatically appended.
Location	Select a folder location for the file in your repository folder.
Top Folder Path (Optional)	This is a subfolder within the project file location where extracted content is placed.
Default Zip File Name (Optional)	The value entered in this field is the default file name when extracting the project to a zip file. When you enter a file name, .zip will be automatically appended when you extract the XF Project item.

3. Select the **Add Project Item** button.
4. In the Project Items grid, configure project items:

Property	Description
Type	This drop-down menu lists all XF Project item types. To view all item types, see Appendix B: XML Project Item Configuration and Compatibility. This property selection filters the other property options and determines whether they are applicable. For example, selecting a business rule type disables the Workspace and Include Descendants properties and filters the Name property's search drop-down menu to only display business rules in the application.
Workspace	The Workspace associated with the project item. A Workspace selection is required for specific item types. If the item type does not belong to a Workspace, this property is not available. See Appendix B: XML Project Item Configuration and Compatibility. For items that are available at both the application level and in Workspaces, like Cube View groups, leaving the Workspaces property blank sets it at the application level (Default Workspace).

Property	Description
Name	This drop-down menu lists all items of the selected type in your application. Depending on your current platform version and the item type selected, this property may not display application items. See Appendix B: XML Project Item Configuration and Compatibility. If the drop-down menu does not display items in your application, you must manually enter the name. **NOTE:** The same name can exist at the application level and inside one or more Workspaces at the same time. When that occurs, the Workspace is what distinguishes them.
Folder Path	The name of the subfolder where the project item type is extracted. If this property is left blank, items are extracted to the default location. If the specified folder path does not exist, it is created during extract.
Include Descendants	This specifies whether child items should be included when processing the project item. For example, when extracting a Workspace and setting Include Descendants to True, all objects contained in that Workspace are also extracted. This property defaults to True. If the item type does not include descendants, like business rules, this property is not available.

> **NOTE:** See Appendix B: XML Project Item Configuration and Compatibility for item types, configuration notes, and compatibility details.



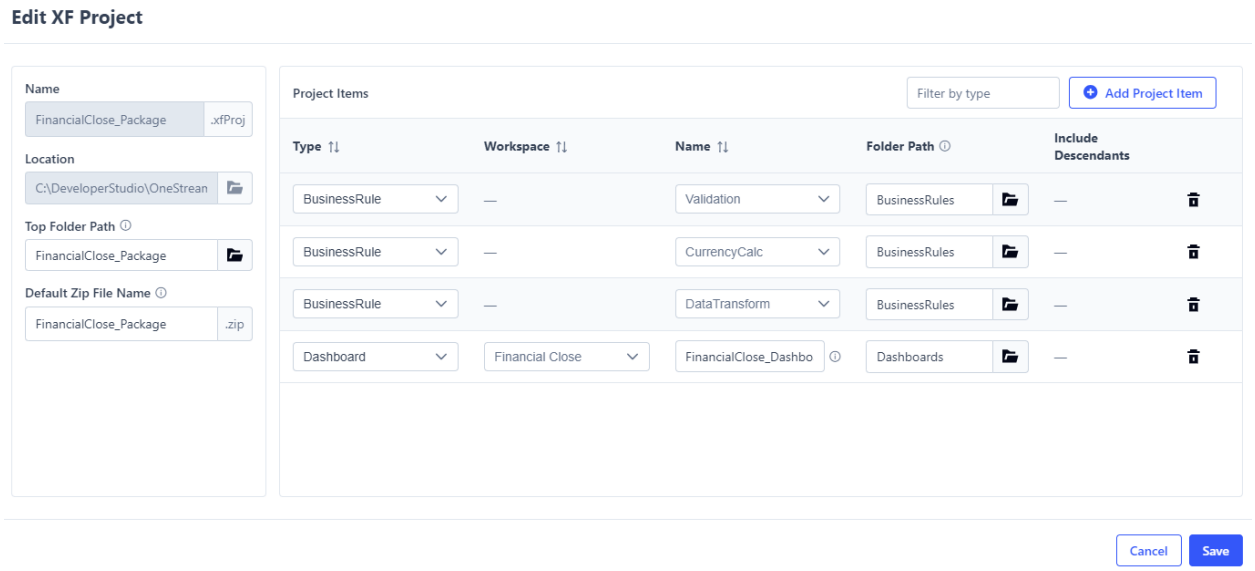
To remove a project item, select the **Remove Project Item** icon.

5. Select the **Save** button.

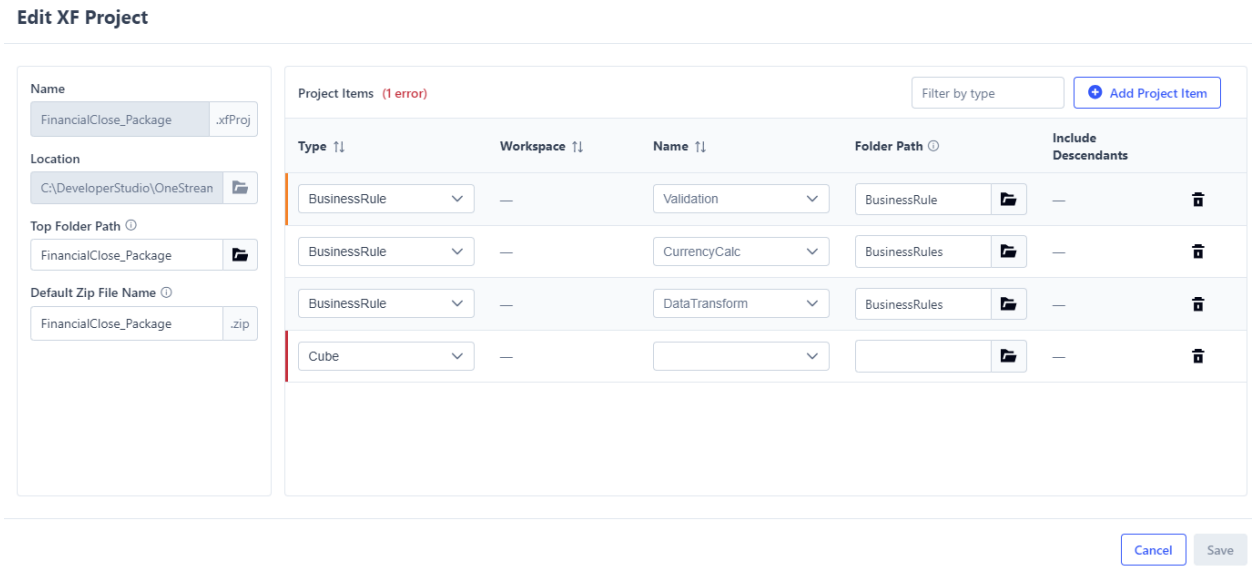
Edit XF Project Files

To edit an XF Project file, hover over the XML item and select the ellipses or right-click on the item to display the context menu. Select **Edit** to display the Edit XF Project window.

> **NOTE:** The Name and Location properties cannot be edited.

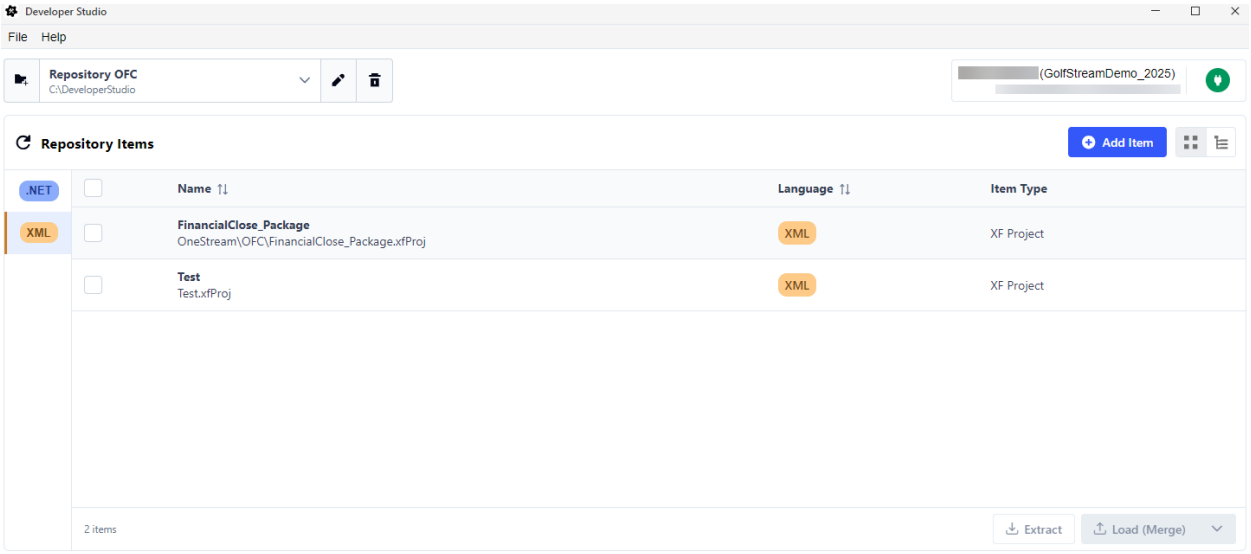


When a project item is edited, the Edit XF Project row displays an orange indicator to show that it has been modified. If a modification is invalid, the row displays a red indicator.

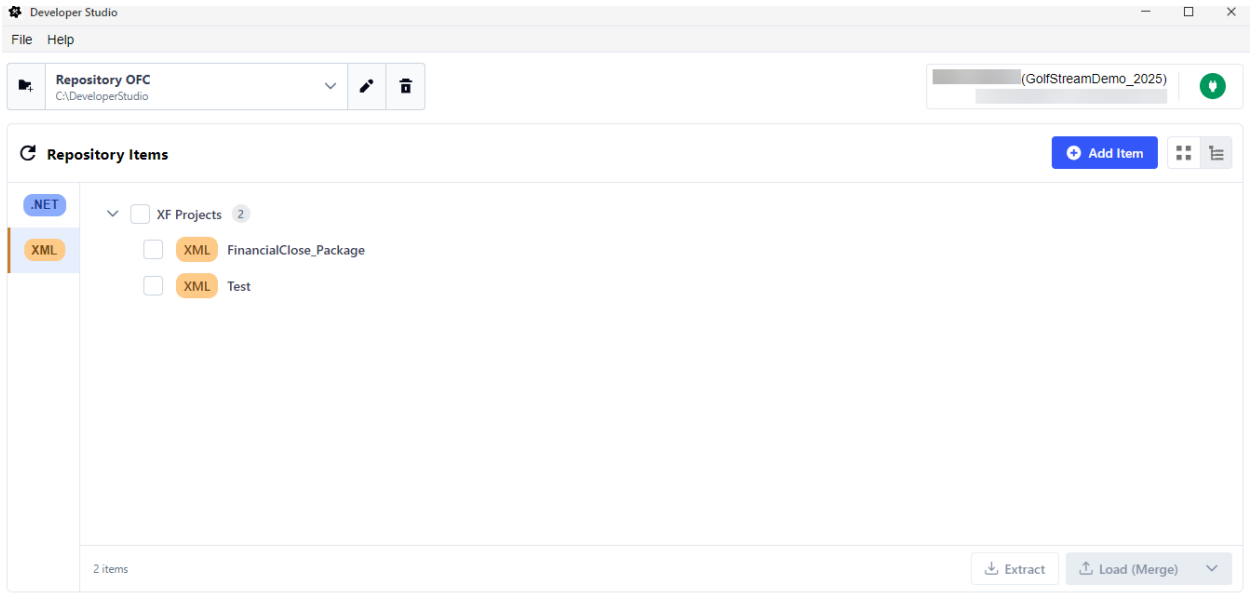


XML Repository Items Grid

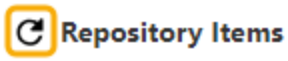
All XML items created display in the XML tab's Repository Items grid.



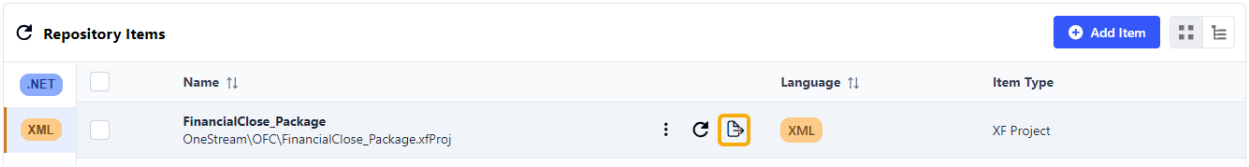
To view items in a hierarchy, select the **Hierarchy View** icon. Select the **Grid View** icon to switch back to a grid view.



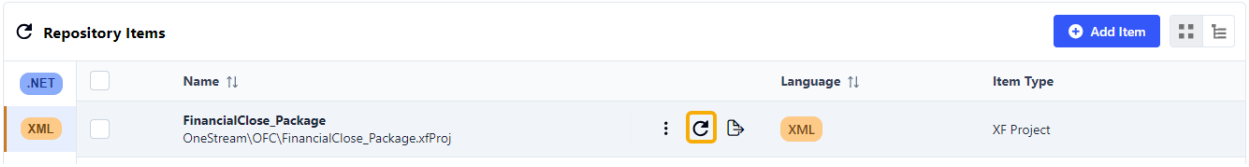
To manually refresh the Repository Items grid, select the **Refresh** icon.



To extract the project to a zip file, select the **Extract to Zip** icon.

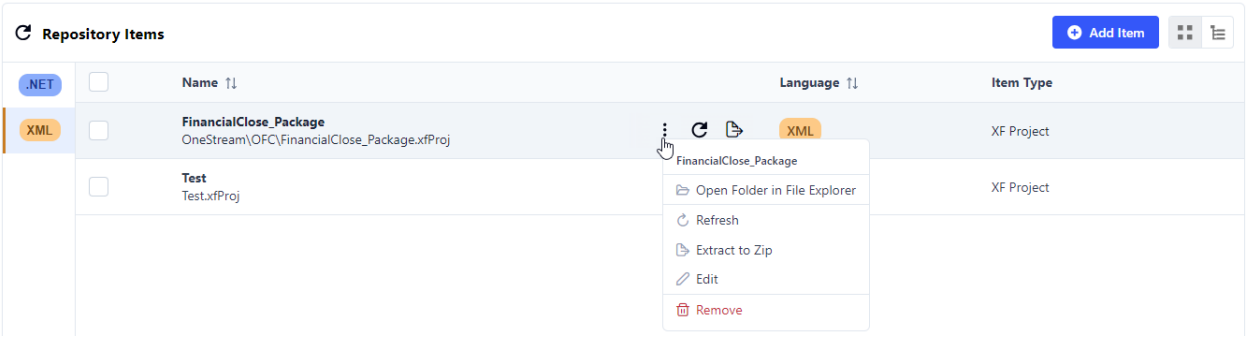


To refresh the XML item, select the **Refresh** icon.



Select the ellipses to display a context menu with the following options:

- **Open Folder in File Explorer**: View the folder location in File Explorer.
- **Refresh**: Refresh the XML item.
- **Extract to Zip**: Extract the project to a zip file.
- **Edit**: Edit the XML item. The Name and Location properties cannot be edited.
- **Remove**: Remove the XML item. When you remove an XML item, nothing is deleted locally or on the server. Deleting your project or source file is done manually outside of Developer Studio.



Load and Extract

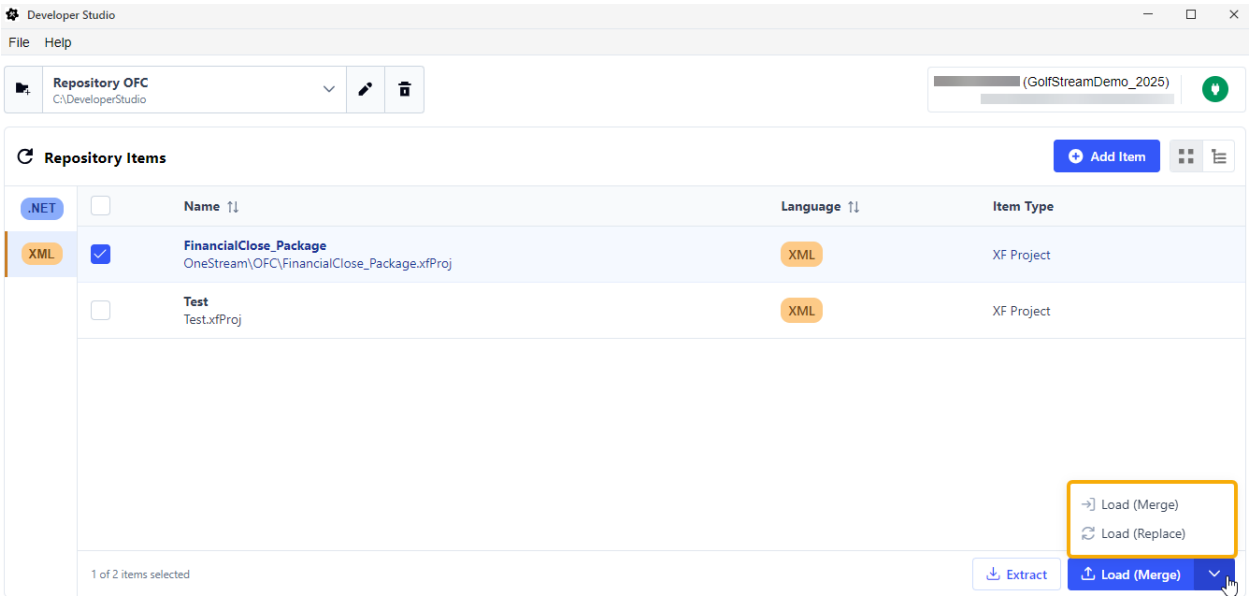
The load operation enables you to load XF Projects from your local to a connected OneStream environment. The extract operation enables you to extract XF Projects from OneStream to your local, either into your local file structure or as a .zip archive. See "XF Projects" in the *Design and Reference Guide*

> **IMPORTANT:** Platform Version 9.4.0 or later is required to perform load and extract operations.

Load

To load an XF Project from your local to OneStream, select the item from the grid or use the multiselect checkbox to perform a bulk load operation. Select one of the following load options:

- **Load (Merge)**: Combines content with existing server artifacts.
- **Load (Replace)**: Replaces existing content with the XF Project.



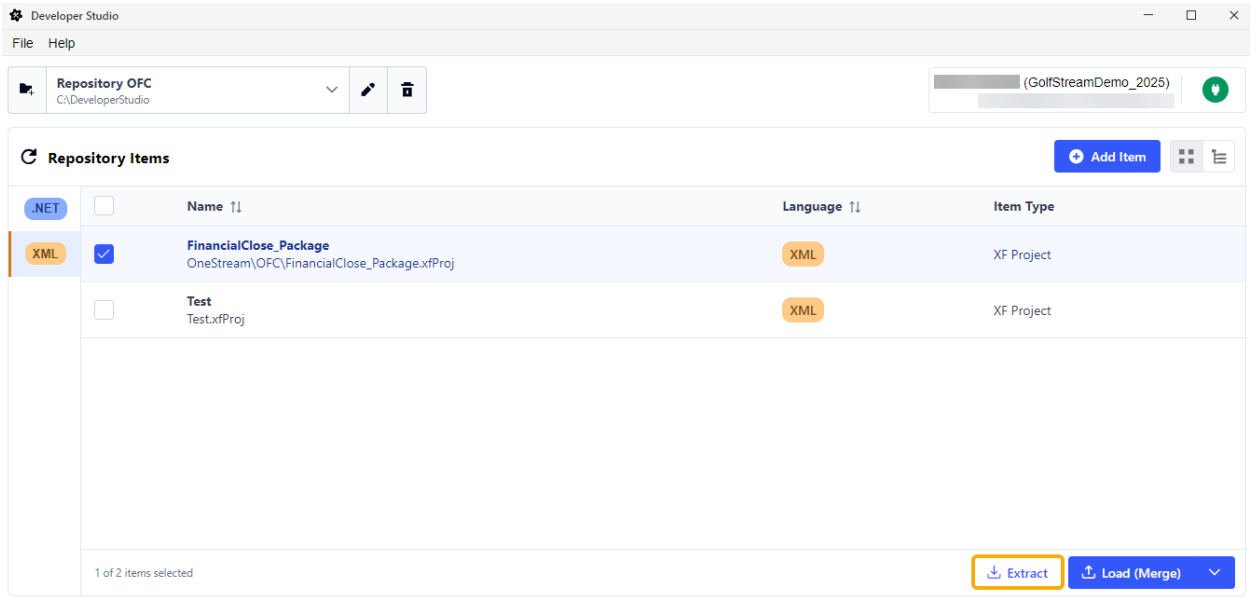
To perform a bulk load, select multiple XML items from the grid using the checkboxes, or select the multiselect checkbox.

When you select a load option, the Confirm Load Operation dialog box displays. To confirm that you want to apply your local content to the OneStream application items, select the **OK** button.

To prevent the Confirm Load Operation dialog box from displaying again, select the **Don't show this confirmation again for load and extract operations** checkbox.

Extract

To extract an XF Project from OneStream to your local, select the item from the grid and select the **Extract** button. To perform a bulk extract operation, select multiple XML items from the grid using the checkboxes or select the multiselect checkbox.



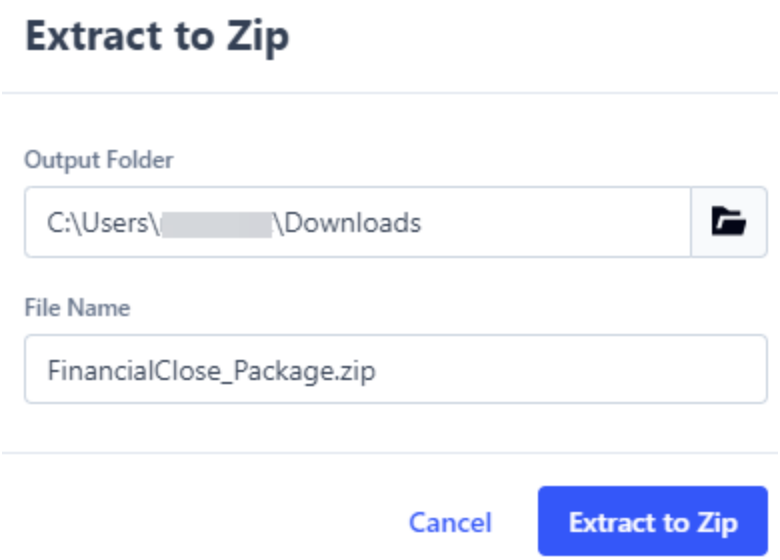
When you select Extract, the Confirm Extract Operation dialog box displays. To confirm that you want to overwrite your local copy of the selected XF Project with the version from the connected OneStream application, select the **OK** button.

To prevent the Confirm Extract Operation dialog box from displaying again, select the **Don't show this confirmation again for load and extract operations** checkbox.

Extract to Zip

To extract to zip, complete the following steps:

1. Select the **Extract to Zip** icon for the XML item, or open the context menu and select **Extract to Zip**. The Extract to Zip dialog box displays.
2. In the **Output Folder** field, select the **Browse** icon and use **File Explorer** to select a folder. This defaults to your Downloads folder.
3. Enter a file name. If the Default Zip File Name property was configured for the item, the field defaults to that value.
4. Select the **Extract to Zip** button.

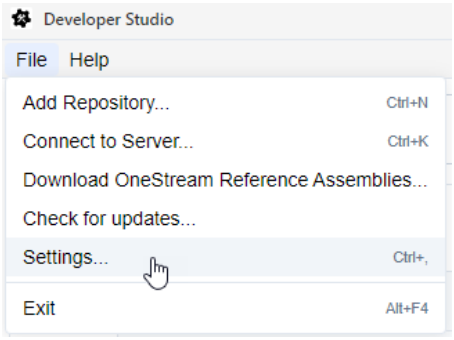


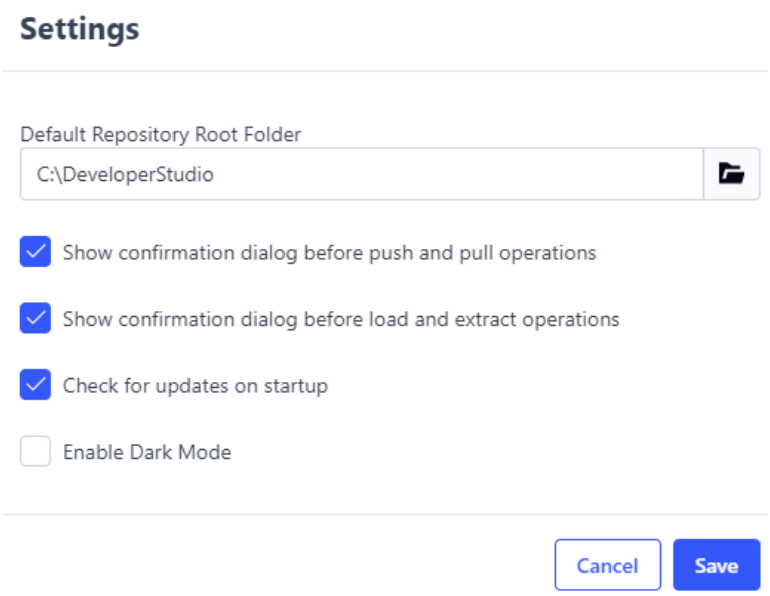
If the file already exists in the selected folder, the File Already Exists dialog box displays. To overwrite the file, select the **Overwrite** button.

> **NOTE:** Bulk extract to zip operations are not supported.

Settings

Navigate to **File** > **Settings**.





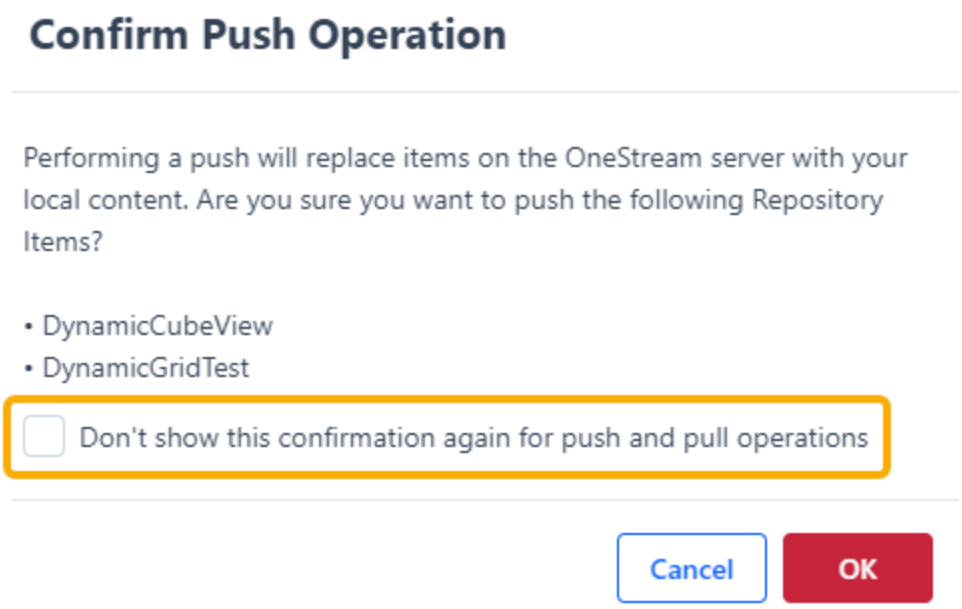
Default Repository Root Folder

To set a default repository root folder, select the **Browse** icon and use **File Explorer** to select a folder. This will display as the default repository root folder setting when you create a new repository.

Show confirmation dialog before push and pull operations

Select this checkbox to display a confirmation dialog before push and pull operations. This is selected by default.

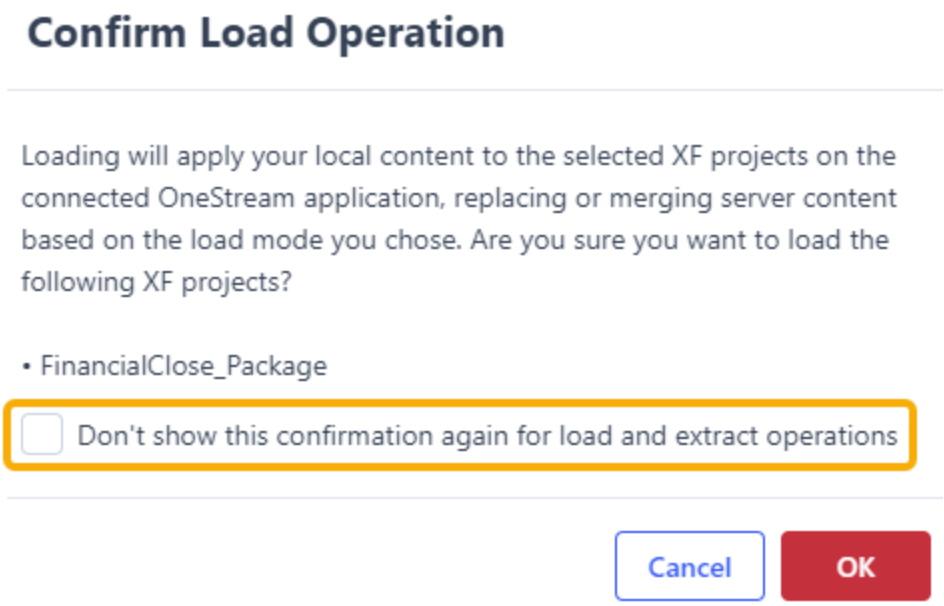
If you previously selected the Don't show this confirmation again for push and pull operations checkbox, selecting this checkbox in Settings restores the confirmation dialog box for those operations.



Show Confirmation dialog before load and extract operations

Select this checkbox to display a confirmation dialog before load and extract operations. This is selected by default.

If you previously selected the Don't show this confirmation again for load and extract operations checkbox during a Load or Extract, selecting this checkbox in Settings restores the confirmation dialog box for those operations.



Check For Updates On Startup

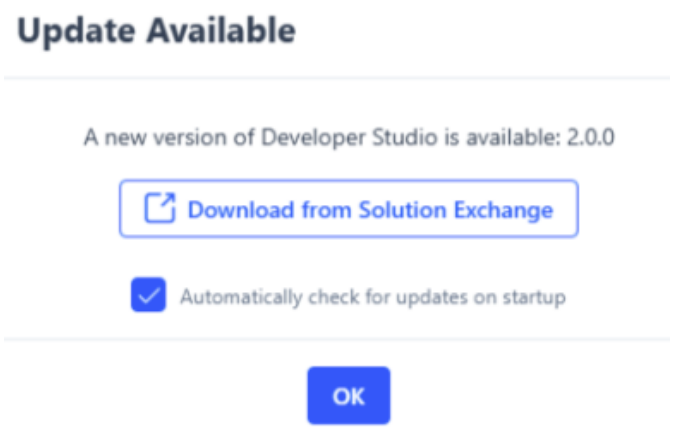
When this checkbox is selected, Developer Studio runs automatic checks on startup to detect available updates. When an update is available, you are directed to Solution Exchange to download the new version. When this checkbox is deselected, automatic checks for updates are not run on startup. This is selected by default. See Automatic Updates.

Enable Dark Mode

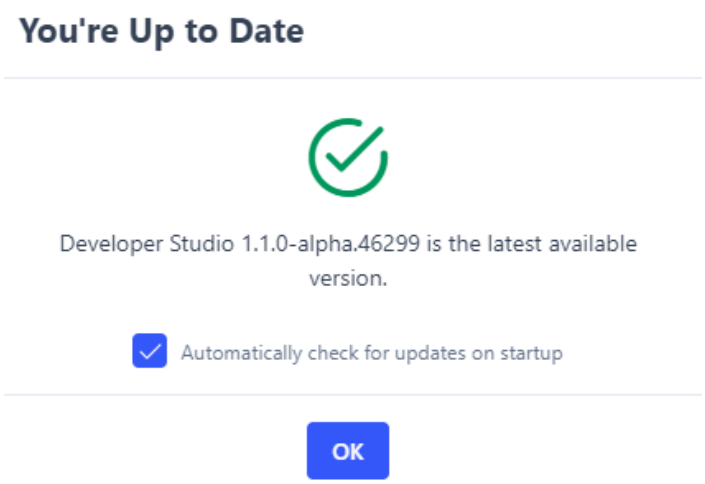
Select this checkbox to enable dark mode.

Automatic Updates

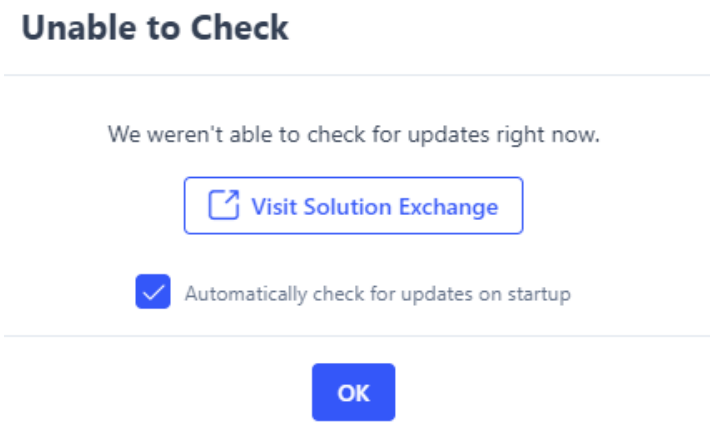
Developer Studio runs automatic checks on startup to detect available updates. When an update is available, the Update Available dialog box displays with the new version number and a link to the Developer Studio page on Solution Exchange where you can access the download link for the new version.



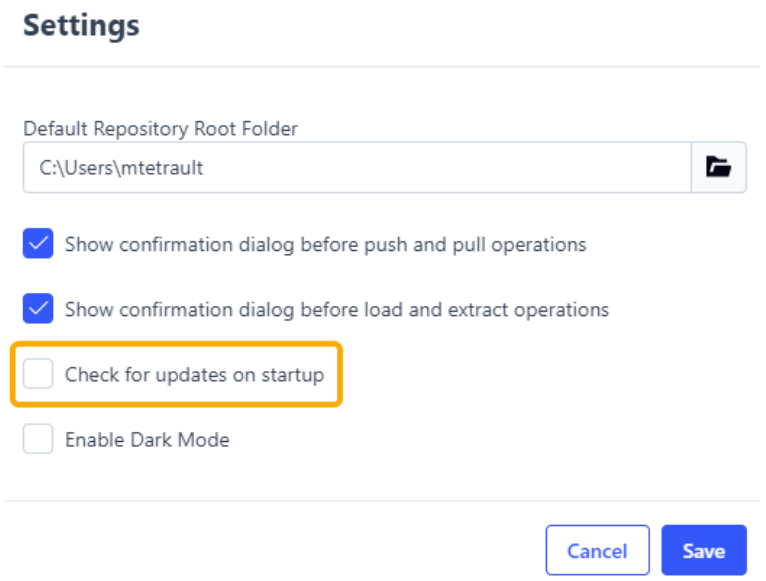
To manually check if a newer version of Developer Studio is available, go to **File** > **Check for Updates**. If an update is available, the Update Available dialog box displays. If you are on the current Developer Studio version, the You're Up to Date dialog box displays.

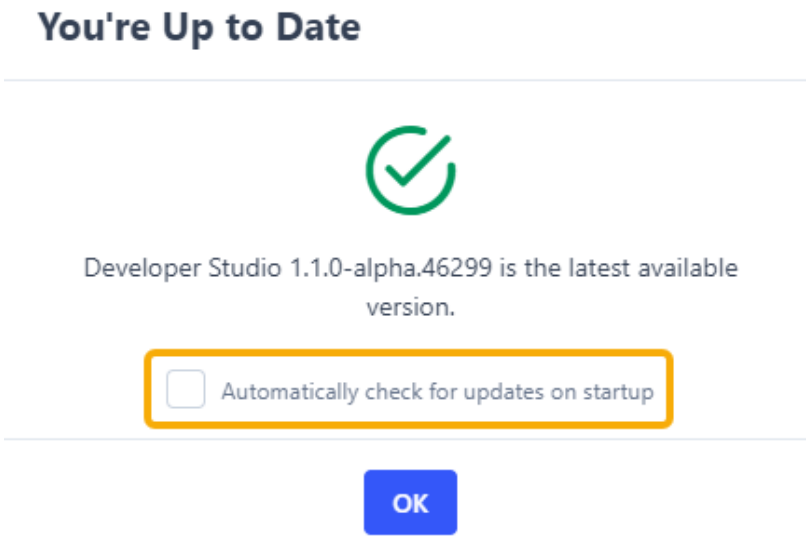


If Check for Updates is selected and the check fails, for example, due to a network error or session timeout, the Unable to Check dialog box displays.



To turn off the automatic update checks on startup, go to **File** > **Settings** or, when manually checking for updates, deselect the **Automatically check for updates on startup checkbox**.


Settings
Default Repository Root Folder
C:\Users\mtetrault
Show confirmation dialog before push and pull operations
Show confirmation dialog before load and extract operations
Check for updates on startup
Enable Dark Mode
Cancel
Save

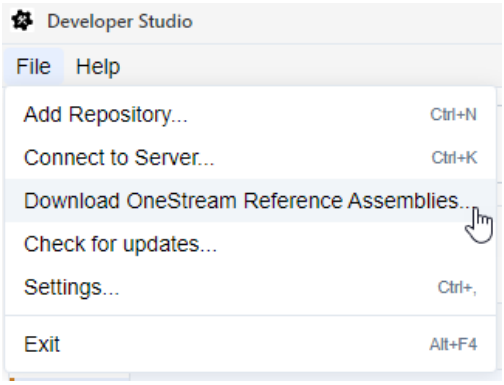

You're Up to Date
Developer Studio 1.1.0-alpha.46299 is the latest available version.
Automatically check for updates on startup
OK

OneStream Reference Assemblies

You can download OneStream reference assemblies to integrate IntelliSense capabilities in your IDE. For information on creating a local NuGet repository, see Setting Up Local NuGet Feeds.

Complete these steps to download reference assemblies and configure IntelliSense:

1. Navigate to **File** > **Download OneStream Reference Assemblies**.



2. From the **Select Version** drop-down menu, select a reference assembly version.
3. Select the **Browse** icon and use **File Explorer** to select the download location and then select the **Download** button. The location defaults to your Downloads folder.
4. When the progress bar is completed, select the **Open Folder in File Explorer** button to display the NUPKG file's folder location.
5. Open a terminal window or command prompt and then navigate to the directory containing the item or project you want to add the reference to. In the following example, the reference is added to the DynamicCubeView project.

```
PS C:\MyRepo\DynamicCubeView> dotnet nuget add source c:\nuget --name LocalRepo
Package source with Name: LocalRepo added successfully.
```

Enter the following command to set up a local NuGet repository for the reference assemblies:

```
dotnet nuget add source
```

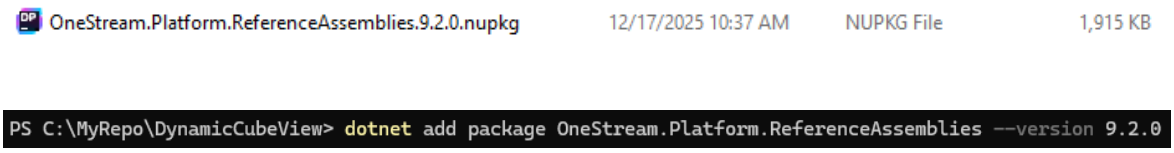
In this example, the repository points to the c:\nuget directory and is named LocalRepo.


```
PS C:\MyRepo\DynamicCubeView> dotnet nuget add source c:\nuget --name LocalRepo
Package source with Name: LocalRepo added successfully.
```


6. Enter this command to add the reference assemblies to your project:

```
dotnet add package OneStream.Platform.ReferenceAssemblies
```

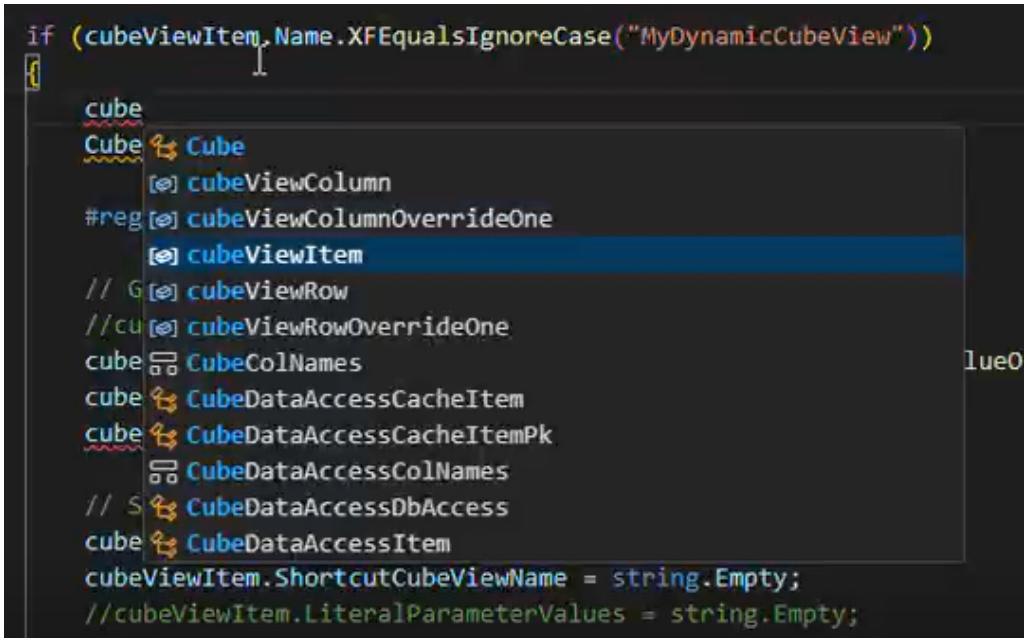
This command must also include the version number listed in the package file name. In this example, the file name lists version 9.2.0, so that version is listed in the command.



> **IMPORTANT:** This second command must be entered in each project file.

For more information on adding dotnet packages and local feeds, see Dotnet Package Add and Setting Up Local NuGet Feeds.

IntelliSense functions in your IDE once these commands are entered. This image shows IntelliSense functioning locally in Visual Studio.


if (cubeViewItem.Name.XFEqualsIgnoreCase("MyDynamicCubeView"))
{
cube
Cube
Cube
cubeViewColumn
cubeViewColumnOverrideOne
cubeViewItem
cubeViewRow
cubeViewRowOverrideOne
CubeColNames
CubeDataAccessCacheItem
CubeDataAccessCacheItemPk
CubeDataAccessColNames
CubeDataAccessDbAccess
CubeDataAccessItem
cubeViewItem.ShortcutCubeViewName = string.Empty;
//cubeViewItem.LiteralParameterValues = string.Empty;

Using the Command Line with Developer Studio

The command line interface (CLI) can be used to perform Developer Studio operations. Actions performed in the CLI are reflected in Developer Studio, enabling you to use your preferred workflow without losing synchronization between tools.

For example, the CLI shares the same server and application context as Developer Studio. If a user connects to a server and opens an application from the CLI, Developer Studio recognizes that connection and updates its context accordingly.

The CLI supports a variety of use cases, including the following scenarios:

- Create, edit, and remove items
- Push and pull changes
- List and filter assemblies or business rules
- Load and extract XF Projects
- Configure settings

To display all available commands, enter `ods` in the CLI. See Command Reference.

```
C:\Users\          >ods
Usage: ods [options] [command]

CLI tool for managing OneStream Developer Studio repository items

Options:
  -V, --version                       Output the CLI version
  --json                              Output machine-readable JSON instead of human text
  -q, --quiet                         Suppress non-essential (status/progress) output
  -v, --verbose                       Enable verbose diagnostic output
  -y, --yes                           Skip confirmation prompts for mutating/destructive commands
  -a, --application <name>            Target application (overrides the stored selection)
  --no-color                          Disable colored output
  -h, --help                          display help for command

Commands:
  connect [options] <serverUrl>       Connect and authenticate with a OneStream server
  disconnect [options]                Disconnect and clear stored credentials
  status [options]                    Show current connection status
  application|app                     Manage OneStream applications
  config                              Manage configuration settings
  repo                                Manage repository definitions in the shared ~/.ods store
  package|pkg                         Download developer packages (e.g. reference assemblies)
  update [options]                    Check whether a newer CLI version is available
  agent                               AI-enablement tools (generate agent skills)
  push [options] [paths...]           Push repository items to the server
  pull [options] [paths...]           Pull repository items from the server
  preview-push [options] [paths...]   Preview what a push would change without uploading
  preview-pull [options] [paths...]   Preview what a pull would change without modifying local files
  validate [options] [paths...]       Validate repository item mapping file(s)
  load [options] <file>               Load an XF project into the connected application
  extract [options] <file>            Extract an XF project from the connected application
  business-rule|br                    Manage business-rule items
  workspace-assembly|assembly         Manage workspace-assembly items
  xf-project|xfproj                   Manage xf-project items
  help [command]                      display help for command
```

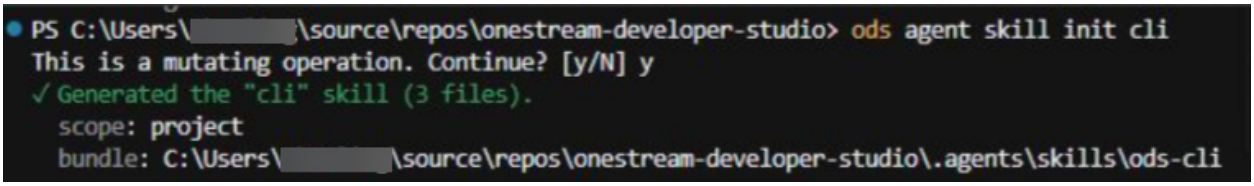
AI Skill File

The CLI also provides compatibility designed for AI-powered development. You can download a skill file directly from the CLI that gives your AI tool contextual knowledge of ODS commands. This enables you to pull down business rules or assemblies into new projects set up with IntelliSense, make changes, and push them back to OneStream all within your AI editor.

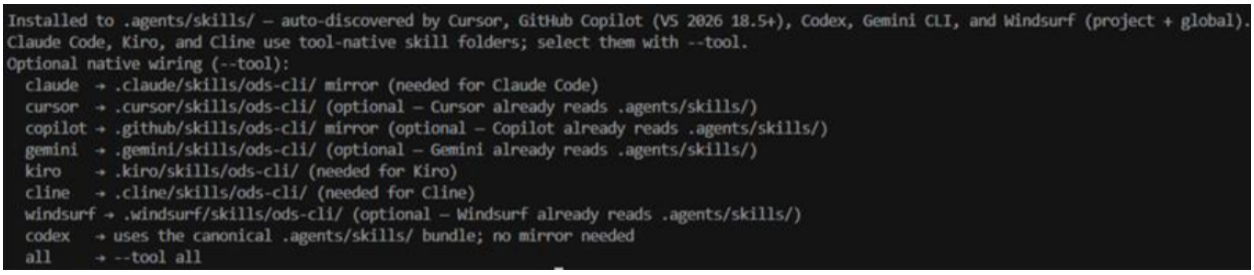
To download the skill file, enter the following code:

```
ods agent skill init cli
```

This command looks for the repository you are currently in. If one is found, it adds or creates the following folder path at the repository root: .agents/skills/ods-cli. If no repository is detected, the folder path is created at the specified folder location.



The `--tool` flag enables you to wire the skill file directly into your editor. The same repository-detection logic applies. However, the destination folder can vary by tool. Some use the same .agents/skills/ods-cli path, while others have their own dedicated folder. The CLI also generates a wrapper alongside the skill file that your editor natively picks up. Once that wrapper is in place, your AI tool automatically knows how to use the skill without any manual configuration.



Command Reference

Use this section to reference skill capabilities, available commands, keywords, expected inputs, and predicted outputs.

Parsing Results and Detecting Failure

Pass `--json` to any command that supports it to get a single, stable envelope on `stdout`:

```json
{
  "schemaVersion": "1.0",
  "command": "<canonical command path>",
  "status": "ok" | "error",
  "data": { /* present when status is ok */ },
  "error": { "code": "<machine code>", "message": "...", "hint": "...", "details": { } },
  "warnings": ["..."]
}
```

Top-Level Commands

ods connect

Connect and authenticate with a OneStream server.

Arguments	<`serverUrl`>. OneStream server URL, for example https://myserver.onestream.com.
Safety	Read. Safe to run autonomously and makes no changes.

```
ods connect https://myserver.onestream.com
```

ods disconnect

Disconnect and clear stored credentials.

Prerequisites	You must be connected first: `ods connect <serverUrl>`

Safety	Read. Safe to run autonomously and makes no changes.

```
ods disconnect
```

ods extract

Extract (pull) the items described by a single .xfproj manifest out of the connected application. By default, the project is expanded to disk in place next to the .xfproj.

> **NOTE:** The `--zip` flag saves a single packaged zip. This operation requires OneStream Platform Version 9.4.0 or later.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`<file>`: the Path to the .xfproj manifest.
Options	`--zip`: save the packaged .zip instead of expanding to disk. `--out <path>`: output directory for the packaged zip . This is `--zip` only and defaults to the .xfproj folder.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.
Supports	`--json`. This emits the standard envelope.

```
ods extract ./MyApp.xfproj
ods extract ./MyApp.xfproj --zip --out ./dist
```

ods load

Load (push) the items described by a single .xfproj manifest into the connected application.

> **NOTE:** This requires OneStream Platform Version 9.4.0 or later.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`<file>`: the Path to the .xfproj manifest.
Options	`--mode <mode>`: load mode. Merge (additive) or replace (deletes server content first). This is required.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.
Supports	`--json`. This emits the standard envelope.

```
ods load ./MyApp.xfproj --mode merge
ods load ./MyApp.xfproj --mode replace --yes
```

ods preview-pull

Compares the server against local items and reports what a pull would change.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.

Arguments	`[paths...]`: mapping files or directories to process. Defaults to the current working directory.
Options	`--diff`: Include inline unified diffs in the text output.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods preview-pull
ods preview-pull . --diff
ods preview-pull --json
```

ods preview-push

Compares local items against the server and reports what a push would change.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`[paths...]`: mapping files or directories to process. Defaults to the current working directory.
Options	`--diff`: Include inline unified diffs in the text output.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods preview-push
ods preview-push . --diff
ods preview-push --json
```

ods pull

Pulls the resolved repository items (.osmap) from the connected application, overwriting local files. The item kind is inferred from each mapping file.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`[paths...]`: mapping files or directories to process. Defaults to the current working directory.
Options	`--diff`: Include inline unified diffs in the text output.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.
Supports	`--json`. This emits the standard envelope.

```
ods pull
ods pull ./src/MyRule
ods pull . --yes
```

ods push

Push the resolved repository items (.osmap) to the connected application. The item kind is inferred from each mapping file.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`[paths...]`: mapping files or directories to process. Defaults to the current working directory.
Options	`--diff`: Include inline unified diffs in the text output.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.
Supports	`--json`. This emits the standard envelope.

```
ods push
ods push ./src/MyRule
ods push . --yes
```

ods status

Shows current connection status.

Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods status
ods status --json
```

ods update

Checks if a newer CLI version is available.

Options	`--check`: check for a newer version without installing (read-only).
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods update --check
ods update --check --json
```

ods validate

Validates the resolved repository items (.osmap) locally.

Arguments	`[paths...]`: mapping files or directories to process. Defaults to the current working directory.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods validate
ods validate ./src/MyRule
ods validate . --json
```

Agent Commands

ods agent skill init

Generates the skill bundle (SKILL.md + references/commands.md + references/xf-project.md) into a target path. The default is the current directory. You can optionally wire a tool entrypoint with `--tool`. The command reference is generated from the CLI itself, so it never drifts from the real command surface.

Arguments	`[name]`: the skill to generate. This defaults to the CLI. `[path]`: the target directory. This defaults to the current working directory.
Options	`--tool <tool>`: wire a tool entrypoint. The options are cursor, claude, copilot, or all.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.
Supports	`--json`. This emits the standard envelope.

```
ods agent skill init
ods agent skill init cli ./my-app --tool cursor
ods agent skill init --tool all
```

ods agent skill list

Lists the named skills available to `ods agent skill init`. With `--json`, this emits the array for scripting and agents.

Alias	`ls`
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods agent skill list
ods agent skill list --json
```

Application Commands

ods application clear

Clears the currently selected application.

Safety	Read. Safe to run autonomously and makes no changes.

```
ods application clear
```

ods application current

Shows the currently selected application.

Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods application current
ods app current --json
```

ods application list

Lists available applications from the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>`
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods application list
ods app list --json
```

ods application use

Selects the current application for push/pull operations.

Prerequisites	You must be connected first: `ods connect <serverUrl>`
Alias	`set`
Arguments	`<name>`: name of the application to use
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods application use GolfStream
ods app use GolfStream
```

Business Rule Commands

ods business-rule add

Creates a new business rule with project and mapping files.

Arguments	`[path]`: project directory or file. This defaults to the current working directory.

Options	`-n, --name <name>`: Name for the repository item. `-l, --language <language>`: programming language. CSharp or VisualBasic. `-e, --existing`: use an existing project instead of creating one. `--assemblyName <assemblyName>`: OneStream assembly or business rule name. `-b, --businessRuleType <businessRuleType>`: business rule type.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods business-rule add -n MyRule
```

ods business-rule decrypt

Decrypts a specific business rule on the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Options	`-n, --name <name>`: OneStream business rule name. `--password-stdin`: read the decryption password from stdin.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods br decrypt -n MyRule
echo "$PASSWORD" | ods br decrypt -n MyRule --password-stdin
```

ods business-rule edit

Edits an existing business rule mapping file.

Arguments	`[path]`: mapping file or directory. This defaults to the current working directory.
Options	`-n, --name <name>`: name of the repository item. This is used to find the <name>.osmap file in a directory. `--assemblyName <assemblyName>`: OneStream assembly or business rule name. `-b, --businessRuleType <businessRuleType>`: business rule type.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods business-rule edit -n MyRule
```

ods business-rule encrypt

Encrypts a specific business rule on the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.

Options	`-n, --name <name>`: OneStream business rule name. `--password-stdin`: read the encryption password from stdin. This is for non-interactive use.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods br encrypt -n MyRule
echo "$PASSWORD" | ods br encrypt -n MyRule --password-stdin
```

ods business-rule get

Gets a specific business rule from the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Arguments	`<name>`: OneStream business rule name.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods br get MyRule
ods br get MyRule --json
```

ods business-rule list

Lists all business rule items. Local mapping files by default or server items with `--remote`.

Aliases	`ls`
Arguments	`[path]`: the path to list local items from. This defaults to the current working directory. This is ignored with `--remote`.
Options	`--remote`: list items on the connected server instead of local mapping files. `--type <type>`: filter server results by business rule type. This requires `--remote`.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods br list
ods br list --json
ods br list --remote
ods br list --remote --json
ods br list --remote --type Finance
```

ods business-rule remove

Removes business rule mapping files.

Aliases	`rm`
Arguments	`[path]`:mapping file or directory. This defaults to the current working directory.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods business-rule remove ./src/MyItem
```

ods business-rule types

Enumerates the business rule types the platform recognizes. This requires an authenticated connection. No application selected is needed.

Prerequisites	You must be connected first: `ods connect <serverUrl>`
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods br types
ods br types --json
```

Config Commands

ods config get

Gets a config value by key.

Arguments	`<key>`: config key.
Safety	Read. Safe to run autonomously and makes no changes.

```
ods config get authorityUrl
```

ods config list

Lists all config key-value pairs.

Safety	Read. Safe to run autonomously and makes no changes.

```
ods config list
```

ods config server list

Lists all server-specific configurations.

Safety	Read. Safe to run autonomously and makes no changes.

```
ods config server list
```

ods config server remove

Removes all configuration from a server.

Arguments	`<serverUrl>`: server URL to remove config for.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods config server remove https://localhost:44358
```

ods config server set

Sets a server-specific config value.

Arguments	`<serverUrl>`: server URL. `<key>`: config key (authorityUrl, disableSslVerification). `<value>`: config value.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods config server set https://localhost:44358 disableSslVerification true
```

ods config set

Sets a config value by key. The value is parsed as JSON when possible, so booleans/numbers are stored typed, for example, `true` is converted to boolean true. Anything that is not valid JSON is stored as a string. To set a nested object, pass a JSON object literal as the value. This replaces the whole key (no deep merge), so update one field by reading the current value, editing it, and setting the whole object back.

Arguments	`<key>`: config key. `<value>`: config value. This is parsed as JSON when valid, otherwise as a string.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods config set authorityUrl https://idp.example.com
ods config set enableDarkMode true
ods config set assembliesApiTimeoutSeconds 120
ods config set someObject '{"nested":{"field":1}}'
```

Package Commands

ods package get

Downloads a package by type. For reference assemblies, this downloads the newest versions by default (or a pinned `[version]`) as a .nupkg into the current directory, or `--output <dir>`. With the `--json` flag, this emits the downloaded package metadata including the written path.

Arguments	`<type>`: package type. For example, reference assemblies. `[version]`: specific version to download. This defaults to the newest version.
Options	`-o, --output <dir>`: destination directory. This defaults to the current working directory.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods package get reference-assemblies
ods package get reference-assemblies 9.2.0
ods package get reference-assemblies --output ./libs
```

ods package list

With no argument, this lists the supported package types. With the type, this lists the versions available on the CDN with the newest first. With `--json`, this emits the array for scripting and agents.

Arguments	`[type]`: package type. For example, reference assemblies.

Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods package list
ods package list reference-assemblies
ods package list reference-assemblies --json
```

Repo Commands

ods repo add

Adds a repository definition to the shared ~/.ods/repositories store. An ID is generated automatically. A path that does not exist is accepted with a warning, matching the desktop client.

Arguments	`<name>`: repository display name. `<path>`: absolute path to the repository folder.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.
Supports	`--json`. This emits the standard envelope.

```
ods repo add "My Repo" C:\repos\my-repo
```

ods repo edit

Updates a repository definition by ID or name. Provide `--name` and/or `--path`. At least one of these is required.

Arguments	`<repo>`: repository ID or name.
Options	`-n, --name <name>`: new display name. `-p, --path <path>`: new repository folder path.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.
Supports	`--json`. This emits the standard envelope.

```
ods repo edit "My Repo" --name "Renamed Repo"
ods repo edit 1a2b3c --path C:\repos\moved
```

ods repo list

Lists repository definitions from the shared ~/.ods/repositories store, the same list the desktop application names. With `--json`, this emits the array for scripting and agents, for example, resolve a path, then `cd` and `push`.

Aliases	`ls`
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods repo list
ods repo list --json
```

ods repo remove

Removes a repository definition by ID or name. This only removes the definition from the shared store. It never deletes the files on disk.

Aliases	`rm`
Arguments	`<repo>`: repository ID or name.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.
Supports	`--json`. This emits the standard envelope.

```
ods repo remove "My Repo"
ods repo rm 1a2b3c --yes
```

Workspace Assembly Commands

ods workspace-assembly add

Creates a new Workspace assembly with project and mapping files.

Arguments	`[path]`: project directory or file. This defaults to the current working directory.

Options	`-n, --name <name>`: name for the repository item. `-l, --language <language>`: programming language. CSharp or VisualBasic. `-e, --existing`: use an existing project file instead of creating one. `--assemblyName <assemblyName>`: OneStream assembly or business rule name. `-w, --workspaceName <workspaceName>`: Workspace name. `-m, --maintenanceUnitName <maintenanceUnitName>`: maintenance unit name.
Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods workspace-assembly add -n MyAssembly
```

ods workspace-assembly decrypt

Decrypts a specific Workspace assembly on the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.

Options	`-w, --workspace <workspace>`: Workspace name. `-n, --name <name>`: assembly name. `-f, --files <files...>`: File names to decrypt. `--password-stdin`: read the decryption password from stdin. This is for non-interactive use.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods assembly decrypt -w W -n N -f File.cs
echo "$PASSWORD" | ods assembly decrypt -w W -n N -f File.cs --password-stdin
```

ods workspace-assembly edit

Edits an existing Workspace assembly mapping file.

Arguments	`[path]`: mapping file or directory. This defaults to the current working directory.
Options	`-n, --name <name>`: name of the repository item. This is used to find the <name>.osmap file in a directory. `--assemblyName <assemblyName>`: OneStream assembly or business rule name. `-w, --workspaceName <workspaceName>`: Workspace name. `-m, --maintenanceUnitName <maintenanceUnitName>`: maintenance unit name.

Safety	Mutating (non-destructive). This may run autonomously. The agent may pass `--yes` to skip the interactive prompt.

```
ods workspace-assembly edit -n MyAssembly
```

ods workspace-assembly encrypt

Encrypts a specific Workspace assembly on the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Options	`-w, --workspace <workspace>`: Workspace name. `-n, --name <name>`: assembly name. `-f, --files <files...>`: File names to encrypt. `--password-stdin`: read the encryption password from stdin. This is for non-interactive use.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods assembly encrypt -w W -n N -f File.cs
echo "$PASSWORD" | ods assembly encrypt -w W -n N -f File.cs --password-stdin
```

ods workspace-assembly files

Gets the files of a specific Workspace assembly from the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Options	`-w, --workspace <workspace>`: Workspace name. `-n, --name <name>`: assembly name.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods assembly files --workspace W --name N
```

ods workspace-assembly get

Gets a specific Workspace assembly from the connected server.

Prerequisites	You must be connected first: `ods connect <serverUrl>` An application must be selected: `ods application use <name>` or pass `--application <name>`.
Options	`-w, --workspace <workspace>`: Workspace name. `-n, --name <name>`: assembly name.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods assembly get --workspace W --name N
```

ods workspace-assembly list

Lists Workspace assembly items. It shows local mapping files by default or server items if you use `--remote`.

Arguments	`[path]`: path to list local items from. This defaults to the current working directory. This is ignored with `--remote`.
Options	`--remote`: list items on the connected server instead of local mapping files. `--workspace <workspace>`: filter server results by Workspace name. This requires `--remote`.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods assembly list
ods assembly list --json
ods assembly list --remote
ods assembly list --remote --json
ods assembly list --remote --workspace MyWorkspace
```

ods workspace-assembly remove

Removes Workspace assembly mapping files.

Aliases	`rm`

Arguments	`[path]`: mapping file or directory. This defaults to the current working directory.
Safety	Destructive. This requires explicit human confirmation. Never auto-pass `--yes`. Preview first where a preview command exists.

```
ods workspace-assembly remove ./src/MyItem
```

XF Project Commands

ods xf-project list

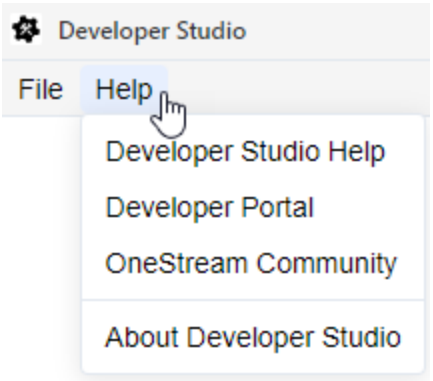
Lists XF Project manifests (.xfproj) on disk.

Aliases	`ls`
Arguments	`[path]`: path to scan for .xfproj manifests. This defaults to the current working directory.
Safety	Read. Safe to run autonomously and makes no changes.
Supports	`--json`. This emits the standard envelope.

```
ods xfproj list
ods xfproj list --json
ods xfproj list ./repo
```

Help Options

Use the Help tab to access support resources.



Developer Studio Help
Access online help documentation.

Developer Portal
Explore the Developer Portal. This is a centralized hub designed to provide developers with resources and guidance.

OneStream Community
Access OneStream Community for forums, collaboration spaces, and additional resources.

About Developer Studio
Access version information and open source licenses.

Appendix A: Third-Party Assemblies

This version of Developer Studio does not support third-party assemblies in the Reference Assemblies NuGet package. If your solution depends on these third-party assemblies for compilation, obtain them through a public NuGet feed:

Assembly	Version Compatible with Developer Studio
AutoMapper	13.0.1
DocumentFormat.OpenXml	3.2.0
HtmlTextWriter	3.0.1
Newtonsoft.Json	13.0.3
Dapper	2.1.35

Appendix B: XML Project Item Configuration and Compatibility

This section details all project item types that can be added to an XF Project file and the capabilities and compatibility details that apply to each type.

The following table defines the columns and values used in the compatibility table:

Column	Description
Item Type	The OneStream content the project item represents.
Workspace	Whether the item belongs to a Workspace. • **Required**: A Workspace must be selected. • **Optional**: The item can be at the application level or inside a Workspace. If the Workspace property is left blank, the item is defined at the application level (Default Workspace). • **Not Used**: The item is not inside a Workspace. Any Workspace value is ignored.

Column	Description
Include Descendants	Whether the Include Descendants property affects what is loaded/extracted for the item. • **Yes**: This is a container. Selecting True includes the items nested beneath it. • **No**: This is a single item. The property has no effect. • **Always**: The item's child files load regardless of the property setting.
Pick From List	Whether Developer Studio can display a list of names to pick from for the selected item type. • **Yes**: You can choose a value from the drop-down menu. • **No**: You must enter the name manually.
Property Requires	The minimum OneStream platform version required for the property to display a drop-down menu of item names. • **X**: No version requirement for this item type. • **Platform Version 9.4.0 or later**: The property is only available for Platform Version 9.4.0 or later. On earlier platform versions, you must enter the name manually.
Notes	End-user details

The following table outlines configuration and compatibility details:

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Business rule	Not used	No	Yes	X	A single rule. Selectable by rule type and name.
Cube	Not used	No	Yes	X	A single cube (metadata).
Cube View group	Optional	Yes	No	X	A container. Setting Includes Descendants to True adds all Cube Views in the group. Enter the name manually.
Cube View	Optional	No	No	X	A single Cube View. Enter the name manually.
Cube View profile	Not used	No	Yes	X	Always application-level and is not tied to a Workspace.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Dashboard Workspace	Required	Yes	Yes	X	A container. Setting Includes Descendants to True adds all its maintenance units and their contents.
Dashboard maintenance unit	Required	Yes	Yes	X	A container inside a Workspace. Setting Includes Descendants to True adds the maintenance unit's content.
Dashboard file	Required	No	No	X	A single file within a Workspace/maintenance unit. Enter the name manually.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Dashboard string	Required	No	No	X	A single string item within a Workspace/maintenance unit. Enter the name manually.
Dashboard parameter	Required	No	No	X	A single item within a Workspace/maintenance unit. Enter the name manually.
Dashboard group	Required	Yes	No	X	A container. Setting Includes Descendants to True adds the child dashboards. Enter the name manually.
Dashboard adapter	Required	No	No	X	A single item within a Workspace/maintenance unit. Enter the name manually.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Dashboard component	Required	No	No	X	A single item within a Workspace/maintenance unit. Enter the name manually.
Dashboard	Required	No	No	X	A single dashboard within a Workspace/maintenance unit/dashboard group. Enter the name manually.
Workspace assembly	Required	Always	Yes	X	This is Workspace-scoped. The assembly's child files always load and extract regardless of what the Include Descendants property is set to.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Dashboard profile	Not used	No	Yes	Platform Version 9.4.0 or later	This is application-level, not Workspace-scoped. The Name drop-down menu requires Platform Version 9.4.0 or later to display values.
Data management group	Optional	Yes	No	X	A container. Setting the Includes Descendants setting to True adds the item's steps/sequences. Enter the name manually.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Data management step	Optional	No	No	X	A single step. Steps are unique only within their data management group, so standalone selection is limited. Enter the name manually. **CAUTION:** If you attempt to extract a data management step without its parent group or sequence, the extract will fail.

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Data management sequence	Optional	No	No	X	A single sequence. Enter the name manually.
Data source	Not used	No	Yes	Platform Version 9.4.0 or later	A single data source. This is not Workspace-scoped. The Name drop-down menu requires Platform Version 9.4.0 or later to display values.
Dimension	Not used	No	Yes	X	A single dimension (metadata).

Item Type	Workspace	Include Descendants	Pick From List	Property Requires	Notes
Transformation rule group	Not used	No	Yes	Platform Version 9.4.0 or later	A single item. This is not Workspace-scoped. This item type has no XF Project descendants. The Name drop-down menu requires Platform Version 9.4.0 or later to display values.
Transformation rule profile	Not used	No	Yes	Platform Version 9.4.0 or later	A single item. This is not Workspace-scoped. The Name drop-down menu requires Platform Version 9.4.0 or later to display values.
Task Scheduler	Not used	No	No	X	A single item. This is not Workspace-scoped. Enter the name manually.

> **NOTE:** The same name can exist at the application level and inside one or more Workspaces at the same time. When that occurs, the Workspace is what distinguishes them.