Adding Libraries

The **Service** panel shows all libraries available in the service. Use the panel to manage your libraries and to use its contents during modeling on the execution pane.

You have two options to add a library to a service:

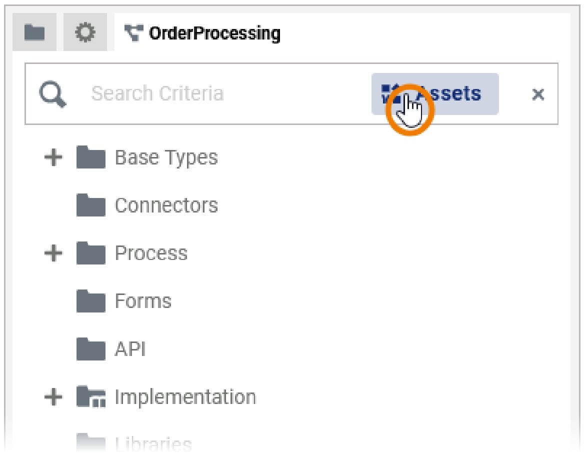
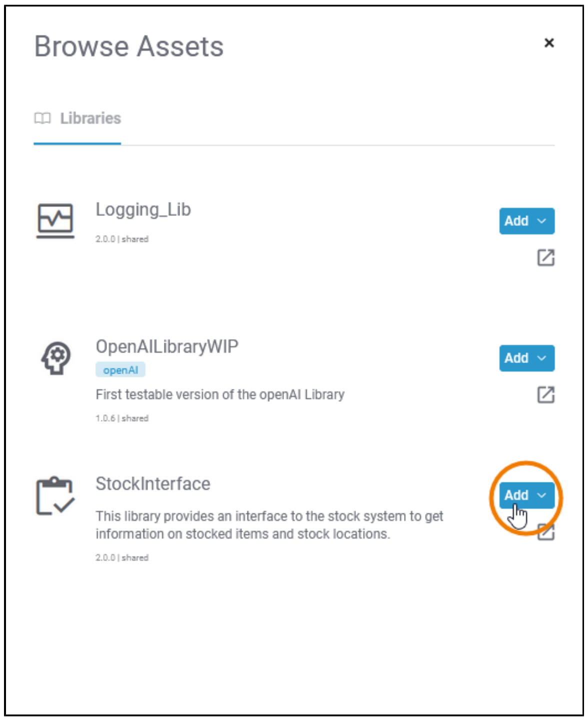
- via the Asset Drawer
- via the Manage Libraries Dialog

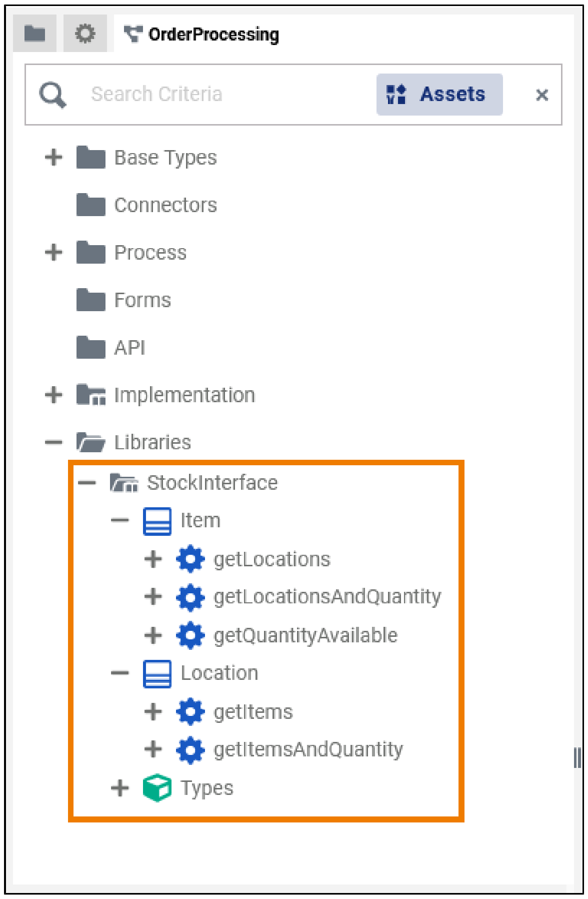
On this Page:

Related Pages:

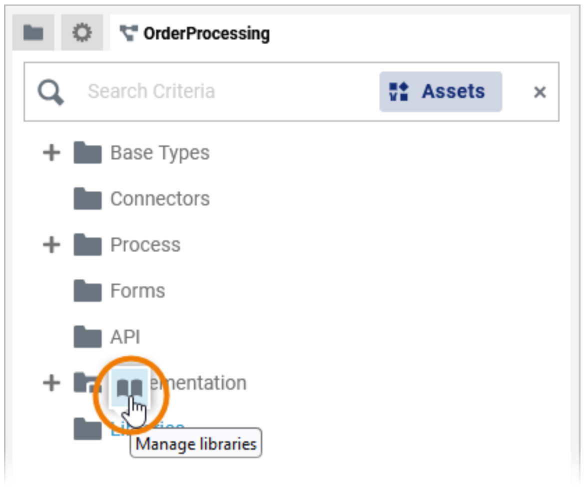
- Working With Libraries
  - Updating Libraries
  - Removing Libraries
- Sharing Designer Content
  - Working With the Asset Drawer
- PAS Designer Administration
  - Administrating Libraries
- PAS Designer Developer Guide
  - Developing and Using Libraries
  - Creating a Library

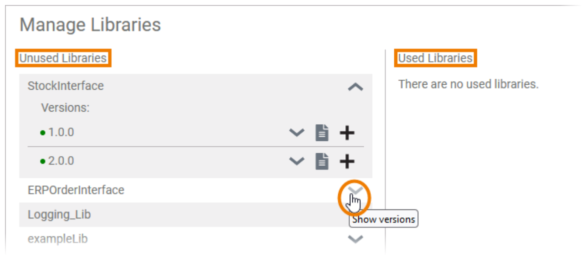
Adding a Library via the Asset Drawer

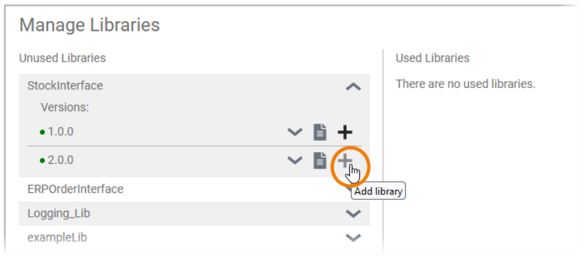
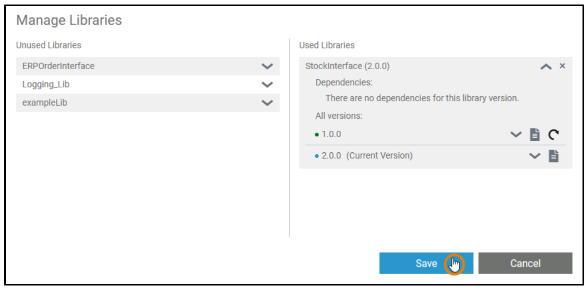
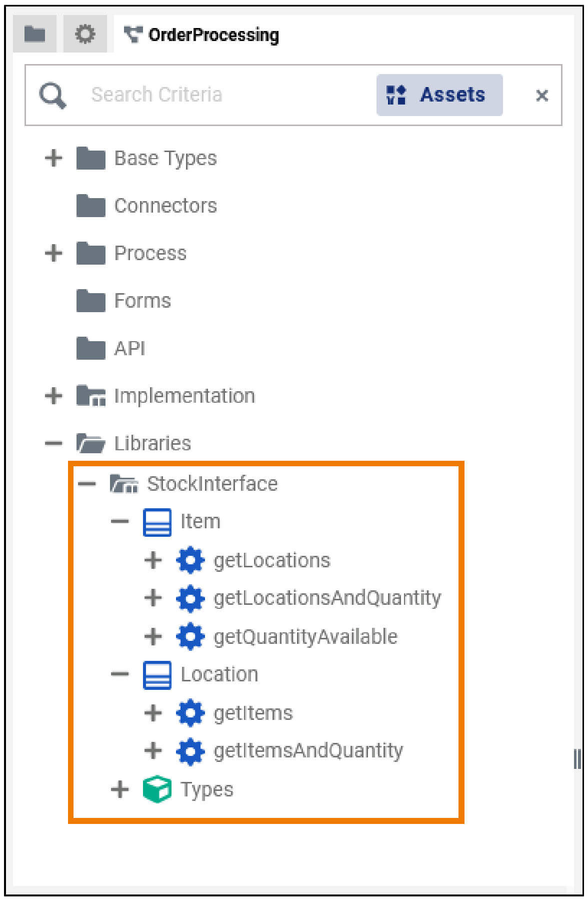
 OrderProcessing Search Criteria Assets Base Types Connectors Process Forms API Implementation Libraries	Open the **Asset Drawer** by clicking button **Assets** in the search box of the service panel.
 Browse Assets Libraries Logging_Lib 2.0.0 | shared Add OpenAILibraryWIP openAI 1.0.6 | shared Add StockInterface 2.0.0 | shared Add First testable version of the openAI Library This library provides an interface to the stock system to get information on stocked items and stock locations.	The Asset Drawer lists all available assets. It displays assets... - ...that have been uploaded to the current namespace. - ...that have been uploaded to namespace **Shared**. Browse the list to find the asset you want to use in your service. To add a library to your service click the corresponding **Add** button.

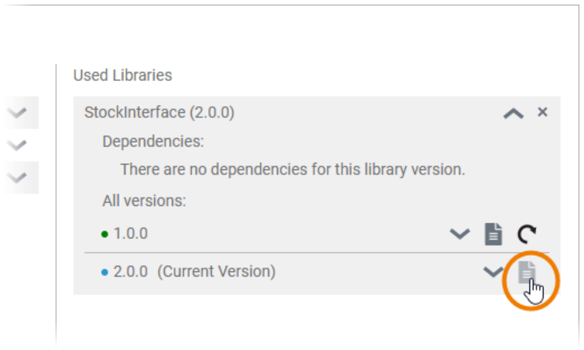
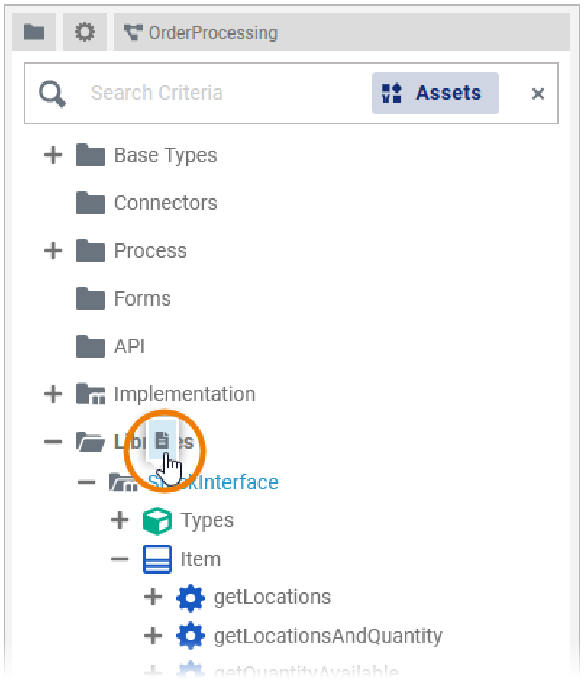
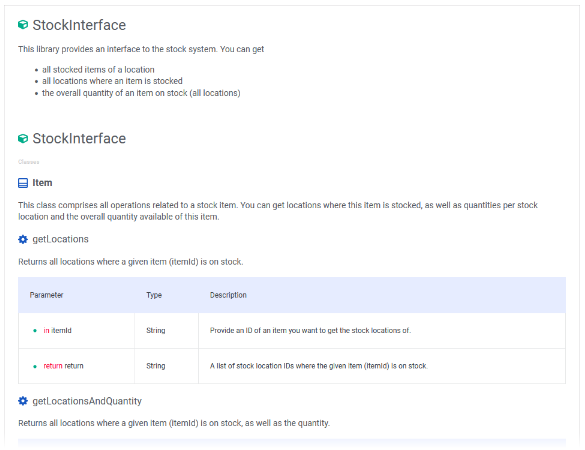
	The added library is now displayed in the service panel and you can use it during modeling.

Adding a Library via the Manage Libraries Dialog

	If you want to add your own libraries to a service, hover over the **Libraries** folder in the service panel and use option **Manage Libraries** .

 Manage Libraries Unused Libraries StockInterface Versions: 1.0.0 2.0.0 ERPOrderInterface Logging_Lib exampleLib Show versions Used Libraries There are no used libraries.	The dialog **Manage Libraries** opens. All libraries that have been uploaded to the current namespace are shown in this dialog: • Currently unused libraries are displayed on the left side of the window. • The libraries that are already used in this service are shown on the right side. Expand the drop-down lists to display the details of each library such as versions and dependencies.

 Manage Libraries Unused Libraries StockInterface Versions: ● 1.0.0 ● 2.0.0 Add library ERPOrderInterface Logging_Lib exampleLib Used Libraries There are no used libraries.	To add an unused library to the project, click **Add library** **+**.
 Manage Libraries Unused Libraries ERPOrderInterface Logging_Lib exampleLib Used Libraries StockInterface (2.0.0) Dependencies: There are no dependencies for this library version. All versions: ● 1.0.0 ● 2.0.0 (Current Version) Save Cancel	The library is now displayed in the column **U sed Libraries**. If you have finished, click **Save** to persist your changes.
 OrderProcessing Search Criteria Assets Base Types Connectors Process Forms API Implementation Libraries StockInterface Item getLocations getLocationsAndQuantity getQuantityAvailable Location getItems getItemsAndQuantity Types	The added library is now displayed in the service panel and you can use it during modeling.

	Use option **Documentation** to display the documentation of a library. You can find it in the **Manage Libraries** dialog...
	... as well as in the service panel when you hover over a library.
	If documentation is available, it will be opened in a new browser tab. This allows developers to access library documentation at any time during the development process.

StockInterface

This library provides an interface to the stock system. You can get

- all stocked items of a location
- all locations where an item is stocked
- the overall quantity of an item on stock (all locations)

StockInterface

Classes

Item

This class comprises all operations related to a stock item. You can get locations where this item is stocked, as well as quantities per stock location and the overall quantity available of this item.

getLocations

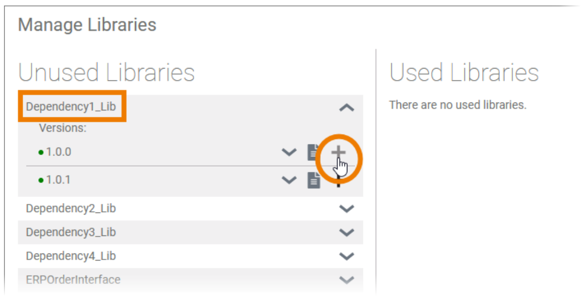
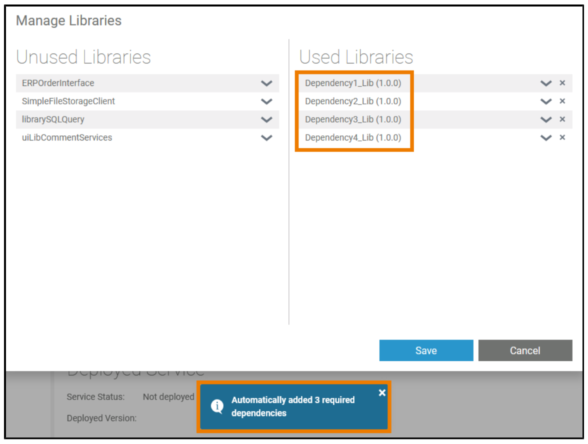
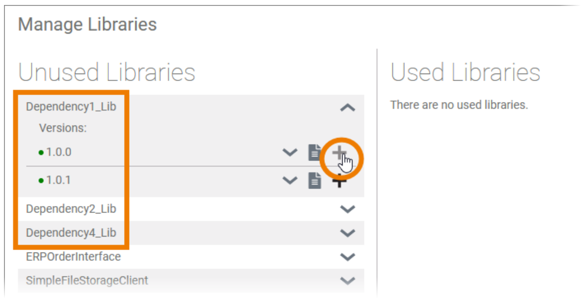
Returns all locations where a given item (itemId) is on stock.

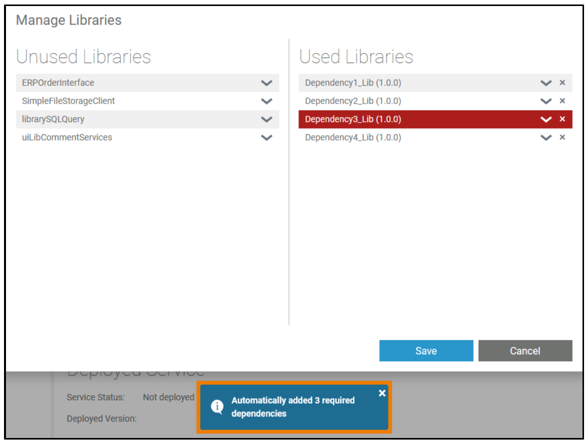
Parameter	Type	Description
• in itemId	String	Provide an ID of an item you want to get the stock locations of.
• return return	String	A list of stock location IDs where the given item (itemId) is on stock.

getLocationsAndQuantity

Returns all locations where a given item (itemId) is on stock, as well as the quantity.

Adding Libraries With Dependencies

 Manage Libraries Unused Libraries Dependency1_Lib Versions: 1.0.0 1.0.1 Dependency2_Lib Dependency3_Lib Dependency4_Lib ERPOrderInterface Used Libraries There are no used libraries.	Libraries may have dependencies between each other. The Designer supports you to add all necessary libraries: When you add a library that contains dependencies...
 Manage Libraries Unused Libraries ERPOrderInterface SimpleFileStorageClient librarySQLQuery uiLibCommentServices Used Libraries Dependency1_Lib (1.0.0) Dependency2_Lib (1.0.0) Dependency3_Lib (1.0.0) Dependency4_Lib (1.0.0) Save Cancel Service Status: Not deployed Deployed Version: Automatically added 3 required dependencies	... the Designer also adds all dependent libraries.
 Manage Libraries Unused Libraries Dependency1_Lib Versions: 1.0.0 1.0.1 Dependency2_Lib Dependency4_Lib ERPOrderInterface SimpleFileStorageClient Used Libraries There are no used libraries.	If you add a library, that contains dependencies but not all dependent libraries are available in the current namespace...



Automatically added 3 required dependencies

... the missing library is added, but highlighted in red.

In that case, go to the library administration and upload the missing library to the namespace to make it available in your service.

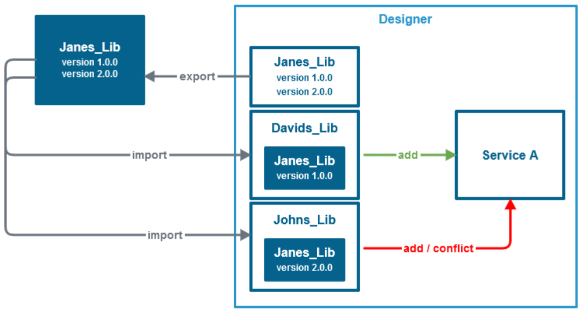
Handling Library Conflicts

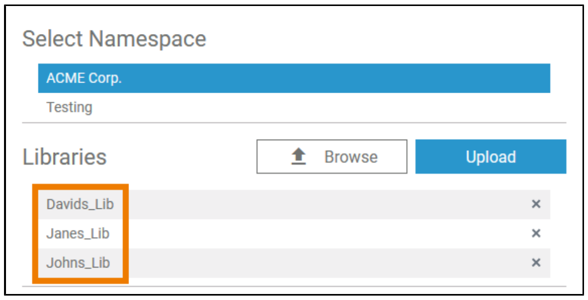
Especially if you develop your own libraries (see Developing and Using Libraries for details) and create different versions, it is possible that library dependencies conflict with each other when you add them to a service.

Example:

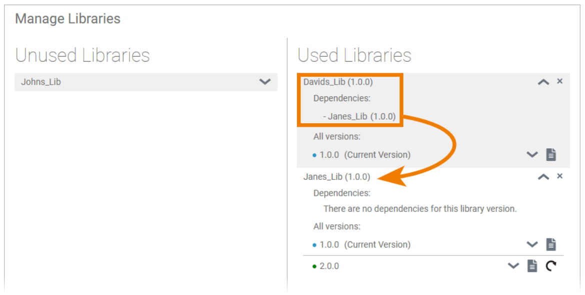
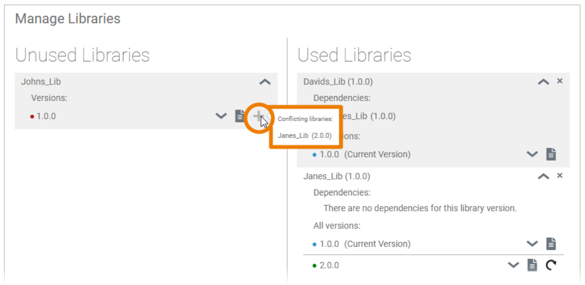
Jane developed her own library, **Janes_Lib** in a first version **1.0.0** . She provides David her library.
He uses **Janes_Lib 1.0.0** during the creation of his own library **Davids_Lib 1.0.0** .

Now, Jane extends her library and creates **Janes_Lib 2.0.0** .
Jane gives the new version of her library to John, who uses **Janes_Lib 2.0.0** in his own library **Johns_Lib 1.0.0** .





All three developers upload their own libraries to a shared namespace, because they want to develop a service together.

 Manage Libraries Unused Libraries Johns_Lib Used Libraries Davids_Lib (1.0.0) Dependencies: - Janes_Lib (1.0.0) All versions: • 1.0.0 (Current Version) Janes_Lib (1.0.0) Dependencies: There are no dependencies for this library version. All versions: • 1.0.0 (Current Version) • 2.0.0	David starts modeling the service. He wants to use his library, so he adds **Davids_Lib 1.0.0** to the service. Since he used **Janes_Lib 1.0.0** in his own library, **Janes_Lib** is also added to the service - in the necessary version **1.0.0** .
 Manage Libraries Unused Libraries Johns_Lib Versions: • 1.0.0 Conflicting libraries: Janes_Lib (2.0.0) Used Libraries Davids_Lib (1.0.0) Dependencies: es_Lib (1.0.0) ons: • 1.0.0 (Current Version) Janes_Lib (1.0.0) Dependencies: There are no dependencies for this library version. All versions: • 1.0.0 (Current Version) • 2.0.0	John wants to collaborate with David, he also wants to use his library **Johns_Lib 1.0.0** . However, the button to add the library is disabled, and John gets a conflict message. Since John used **Janes_Lib 2.0.0** in his own library, there is a version conflict between **Janes_Lib 1.0.0** in Davids library and **Janes_Lib 2.0.0** in Johns library.

John and David have now two possibilities to solve the version conflict:

- David can upgrade the library version of Janes library to use **Janes_Lib 2.0.0** in his library. This is probably the most common case.
- John could downgrade the library version of Janes library to use **Janes_Lib 1.0.0** in his library.

Once David and John are using the same version of **Janes_Lib** , both can use their libraries in the service.