

Working With Libraries

The Library Concept

Libraries are code repositories that are useful to organize your development project into re-usable pieces that can be used in multiple services. They contain predefined classes, interfaces, operations and parameters you can use during modeling by simple drag & drop.

Libraries are developed with the [Builder](#). The Designer comes with a standard library which already provides all necessary [Base Types](#) and base type operations.

Using libraries has several advantages for developers:

- You can provide additional data types and operations via libraries.
- You can reuse implementations you have already developed with the Builder.
- You can use libraries for modularization:
 - Implementations can be recycled in various processes via libraries.
 - Multiple developers can contribute to the same process implementation by working on different libraries.
- You can add features to the Designer, for example access to backend systems, via libraries.
- You can use different versions of the same library as the library administration supports versioning.

Library Usage in Designer

In the Designer, you can create your own libraries, see page [Creating a Library](#) for details. If you want to use the libraries in a Designer service, you have to upload them to a namespace. This is done in the **Libraries** section of the [Designer administration](#). The uploaded libraries are then available in all services created in this namespace. Since PAS 23.1.1 it is also possible to share a library with all users on your PAS installation. Refer to [Sharing Designer Content](#) for details.

How to upload your libraries to the Designer is explained in detail on page [Adminstrating Libraries](#).

Once a library has been uploaded, you can add it to any service created in this namespace. To do so,

open the service, click the  **Assets** button and use the [Asset Drawer](#) to add them or go to the **Libraries** folder in the **Service** panel and use its context menu.



Go to page [Adding Libraries](#) for detailed information on how to add a library to a service. Refer to chapter [Modeling Execution](#) for further information about the usage of libraries during modeling.

On this Page:

- [The Library Concept](#)
 - [Library Usage in Designer](#)
- [The Service Panel](#)

Related Pages:

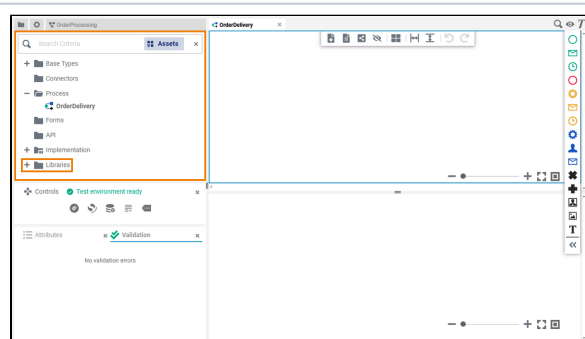
- [Adding Libraries](#)
- [Updating Libraries](#)
- [Removing Libraries](#)
- [Sharing Designer Content](#)
 - [Publishing Assets](#)
- [PAS Designer Administration](#)
 - [Adminstrating Libraries](#)
- [PAS Designer Developer Guide](#)
 - [Developing and Using Libraries](#)
 - [Creating a Library](#)
 - [Using a Library](#)

Related Documentation:

- [BRIDGE](#)
 - [BUILDER User's Guide](#)

The Service Panel

The libraries you have added to a service reside in the **Service** panel.



If you use the panel preset of the BPMN editor, the **Service** panel is displayed in the upper left corner of the editor.



Refer to [Customizing Editors and Panels](#) for detailed information about panel management in general.

Search Criteria

Assets

+

Base Types

Connectors

+

Process

Forms

API

+

Implementation

-

Libraries

-

StockInterface

-

Types

-

ItemLocation

locationId: String

quantity: Float

-

Item

+

getLocations

+

getLocationsAndQuantity

-

getQuantityAvailable

in itemId: String

return: Float

The **Service** panel gives you direct access to all libraries of your service, and thus to all of your predefined classes, interfaces, operations and parameters.

Open menu **Libraries** in the [Designer administration](#) to upload your own libraries.

A library can contain the following elements:

Icon	Name
	Package
	Class
	Property
	Interface
	Operation
	Parameter

To expand an element in the tree, click on **+**.
Use **-** if you want to collapse a tree below an element.

Implementation

-

Libraries

-

StockInterface

-

Types

+

ItemLocation

+

StockedItem

-

Item

+

getLocations

+

getLocationsAndQuantity

+

getQuantityAvailable

+

Location

getLocations

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

Each library element can have a documentation. Hover your mouse on an element in the libraries folder to see if documentation is available. The documentation is displayed as a tooltip.

	You can also copy & paste content from a library to the data model of your service: packages, classes, properties, operations and parameters - including their subelements. The implementation of a copied operation, however, will be empty.
--	--

In addition to imported libraries with predefined data types, you can use the provided **Base Types**, or you can create your own data model.

	The Designer provides all necessary base types in a Bridge Base standard library. This library is available in all services and cannot be removed. It contains the following xUML base types: • Any • Blob • Boolean • DateTime • Float • Integer • String Most of these base types are only able to hold one single piece of information, like text in a string, true or false in a boolean, or binary data in a blob. ✓ Refer to Available Base Types for more information on the xUML base types. If you want to associate several bits of information, you have to define a complex type that combines a number of independent base types and possibly other complex types. Such complex types are modeled as classes. To use your own types, you can • define your own data structures in the Implementation folder • provide them via a library. ✓ How to upload your own libraries is explained on Designer Administration > Libraries.
--	---

Search Criteria

Assets

+ Base Types

Connectors

- Process

OrderDelivery

Forms

API

- Implementation

+ OrderDelivery

+ Libraries

+ Add

CutCtrl + X

CopyCtrl + C

Rename

DeleteDel

You can also define your own data types in folder **Implementation**. You can add your own packages, classes, interfaces, operations, properties and parameters.

✔

Go to page [Modeling Data Structures](#) for more detailed information on how to work on your own data model and to page [Modeling Data Mapping](#) for further explanations on how to define mappings between data types.

Go to chapter [Developing and Using Libraries](#) to find out how to create your own libraries in the Designer.

You can also define your own data types in folder **Implementation**. You can add your own packages, classes, interfaces, operations, properties and parameters.

✔ Go to page [Modeling Data Structures](#) for more detailed information on how to work on your own data model and to page [Modeling Data Mapping](#) for further explanations on how to define mappings between data types.

Go to chapter [Developing and Using Libraries](#) to find out how to create your own libraries in the Designer.