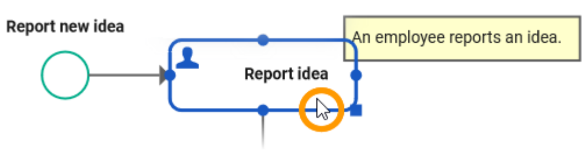
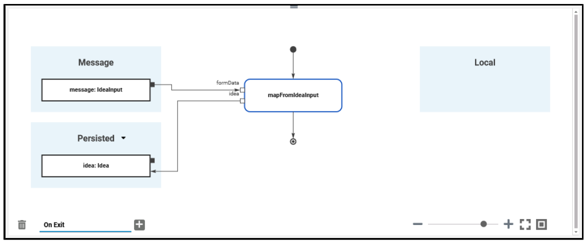
Adding Variables

Adding Variables to the Execution Pane

You need to perform two steps to implement executional parts to your model:

1. Provide all necessary data types and operations for the implementation of your process. These types and operations reside in the **Service** panel of the BPMN editor.
    - You can use the **Base Types** that are provided with the Designer.
    - You can create other necessary types yourself in the **Implementation** section.
      Refer to Modeling Data Mapping for further information.
    - You can import a library that provides additional types and operations.
      Refer to Designer Administration > Libraries for further information.
2. In the second step, select data types and operations from the **Service** panel, and add them to your process at the right places.
    - How this is done will be explained in this chapter.

Select BPMN Element

 Report new idea Report idea An employee reports an idea.	To add executional parts to a BPMN element, you must first select the BPMN element on the diagram pane to which you want to add the execution.
 Message message: IdeaInput formData idea mapFromIdeaInput Local Persisted idea: Idea On Exit	The execution pane displays the related execution model. If there is no execution model in place, you need to create a new execution model (see Working with the Execution Editor > Adding an Execution Model for details).

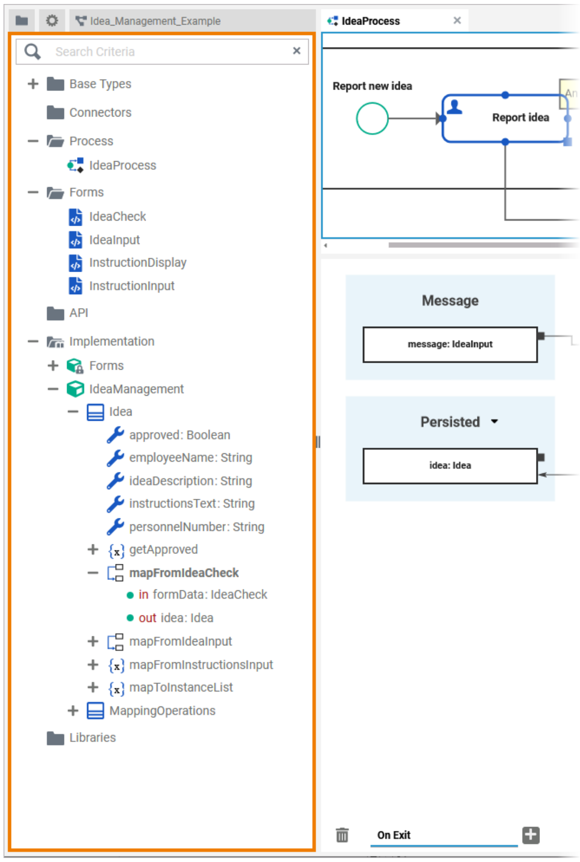
Select Type

On this Page:

Related Pages:

- Service Panel
    - Base Types
- Modeling Execution
    - Working with the Execution Editor
    - Adding Operation Calls
- PAS Designer User Guide
    - Modeling BPMN
    - Working With Libraries
- PAS Designer Administration
    - Administrating Libraries

Next, you need to select the type of the new variable.



Go to the **Service** panel. Types are available from

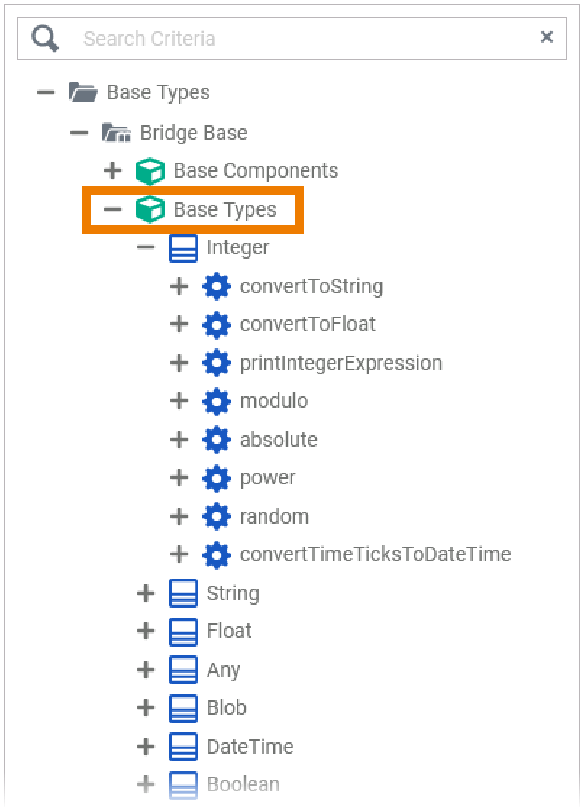
- the standard **Bridge Base Types** library (see below)
- class, interface and parameter definitions from the **Implementation** folder

Go to page Modeling Data Structures for more information on how to define your own classes, operations and interfaces with the Designer.

- imported libraries

- ○ Page Working With Libraries contains more details on the library concept and the **Service** panel.

	- Page Administrating Libraries explains how to upload a library to the Designer.



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.

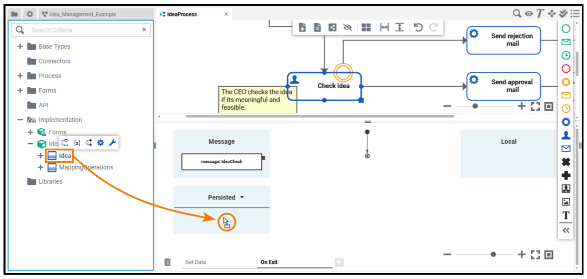
> Refer to Available Base Types for more information on the xUML base types.

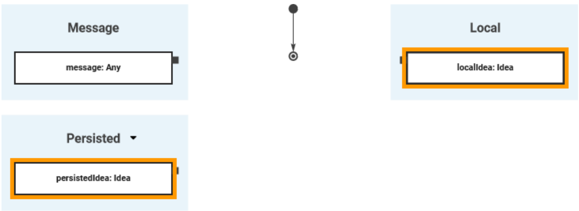
If you want to associate several bits of information, you have to define a complex type that combines a number of independent base types and possibly other complex types. Such complex types are modeled as classes. To use your own types, you can

- 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.

Drag Type to Execution Pane

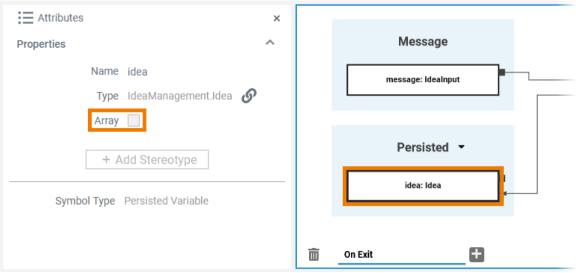
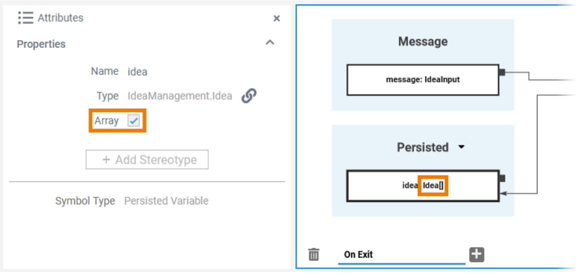
 Idea_Management_Example IdeaProcess Search Criteria Base Types Connectors Process Forms API Implementation Forms Idea Idea MappingOperations Libraries Send rejection mail Check idea Send approval mail The CEO checks the idea if its meaningful and feasible. Message message: IdeaCheck Persisted Local Get Data On Exit	To add a variable to the execution, simply drag the selected type to the execution pane and drop it to a target area. You can cancel this procedure by pressing **Esc** or dropping the element outside the execution pane.

 Message message: Any Local localIdea: Idea Persisted persistedIdea: Idea	You can drop the type to different targets: - to the **Local** section to create a local variable. - to the **Persisted** section, to create a persisted variable. Variables created in section **Persisted** are usable in all executions of the BPMN model. ✓ See page Persisting Data for further information. By dropping a type on an existing variable, you can change the type of the variable (see Changing the Type further below).

Changing a Variable

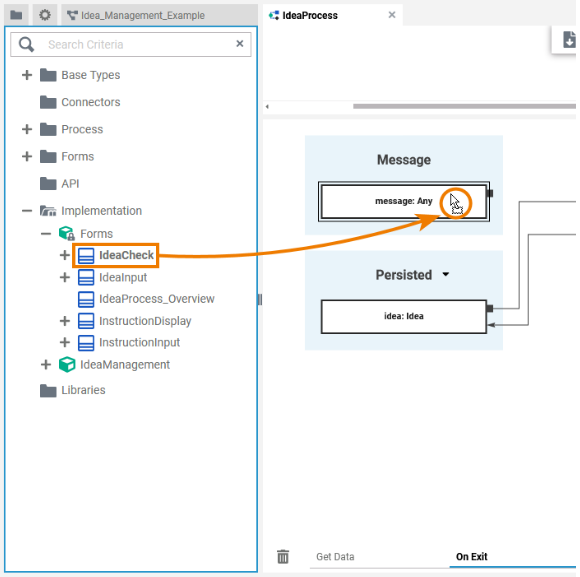
Changing the Multiplicity

As per default, variables are created with multiplicity 0..1 but you can change this in the attributes of the variable.

 Attributes Properties Name idea Type IdeaManagement.Idea Array ☐ + Add Stereotype Symbol Type Persisted Variable Message message: IdeaInput Persisted idea: Idea On Exit	Open the attributes panel for the variable in question. To change the multiplicity to 0..*, activate checkbox **Array**.
 Attributes Properties Name idea Type IdeaManagement.Idea Array ☑ + Add Stereotype Symbol Type Persisted Variable Message message: IdeaInput Persisted idea: Idea[] On Exit	The element type on the pane now is extended by a pair of square brackets to indicate the new multiplicity.

Changing the Type

You can change the type of any variable and there are different ways to do so.

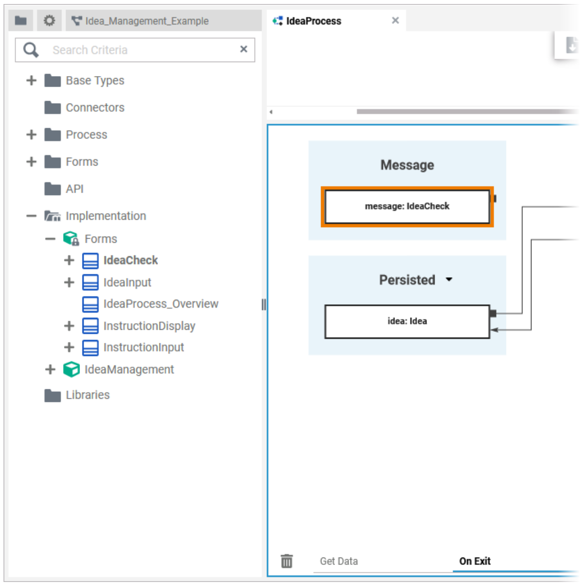


Using Drag & Drop

You can change the type of a variable by dragging a type from your library and dropping it on the variable itself.

Example:

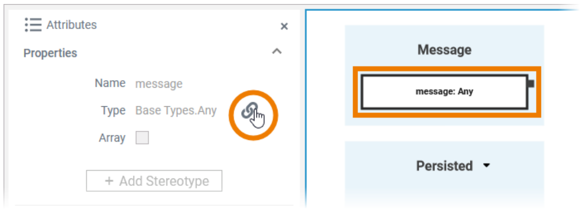
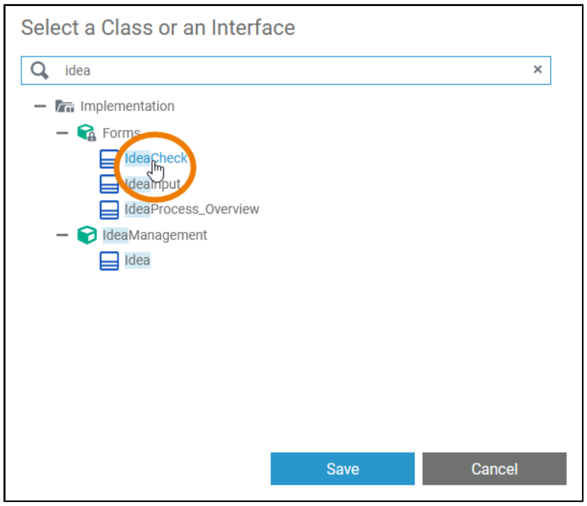
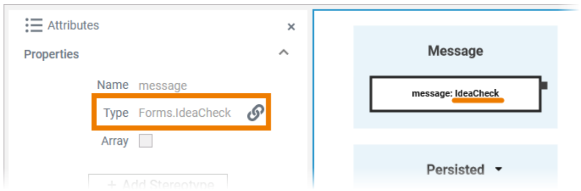
As per default, an incoming message has type **Any**. To gain access to the data you want to use, you need to apply the correct type from your data model or library - which in our example is the form class **IdeaCheck**.



After having dropped the type on the variable, the new type is displayed.

Example:

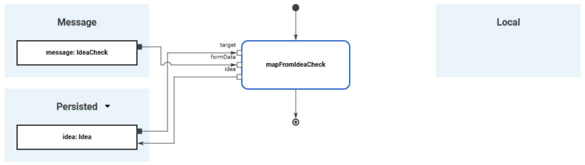
The type of the selected class **IdeaCheck** is applied to the incoming message.

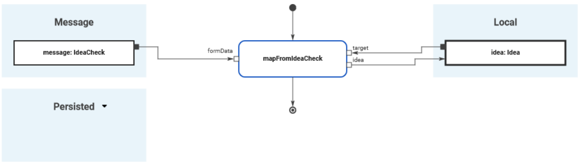
 Attributes Properties + Add Stereotype Message Persisted ▾ Name: message Type: Base Types.Any Array: □ message: Any	**Using the Type Selection Dialog** You can use the **Type Selection** dialog to change the type of a variable. Click on the variable you want to adapt and open the attributes panel. Use icon 🔗 of option **Type** to open the dialog.
 Search: idea Implementation Forms IdeaCheck IdeaInput IdeaProcess_Overview IdeaManagement Idea	Use the search box on top to find the type you are looking for. Click on the type you want to apply. Then click **Save**.
 Attributes Properties + Add Stereotype Message Persisted ▾ Name: message Type: Forms.IdeaCheck Array: □ message: IdeaCheck	The dialog closes and the selected type is applied to the variable.

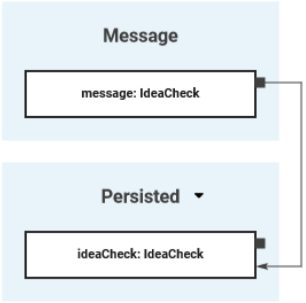
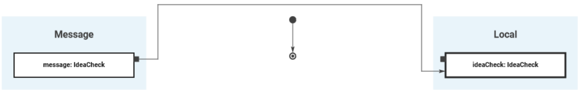
Changing the Scope

It is also possible to change the scope for a variable. You can drag & drop variables:

- **From Local to Persisted**
- **From Persisted to Local**
- **From Message to Persisted or Local**

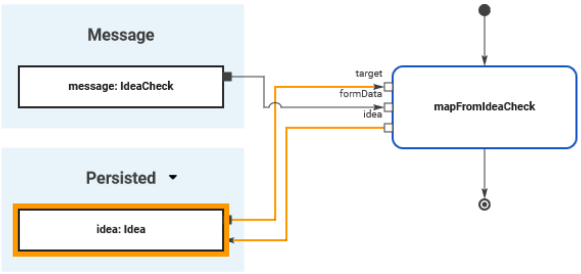
 Message message: IdeaCheck Persisted idea: Idea target formData idea mapFromIdeaCheck Local	**Local to Persisted:** - The local variable is removed. - A new persisted variable with the same name is created. If the name is already in use, a suffix is added. - Existing relations are reattached to the new variable.

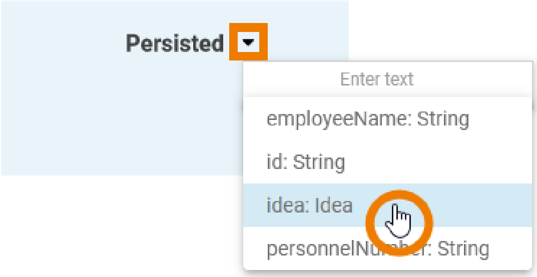
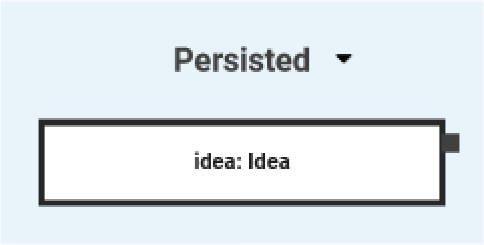
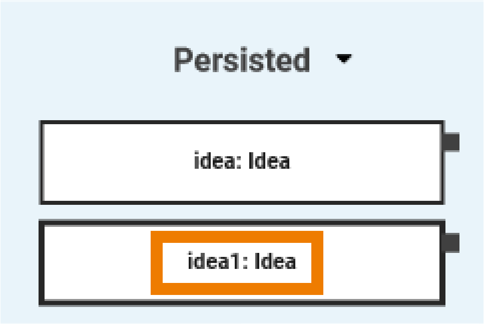
 Message message: IdeaCheck formData mapFromIdeaCheck target idea Local idea: Idea Persisted	**Persisted to Local:** - A new local variable with the same name is created. If the name is already in use, a suffix is added. - The persisted variable is removed if it is not still used in other places in the process. - Existing relations are reattached to the new variable.

 Message message: IdeaCheck Persisted ideaCheck: IdeaCheck  Message message: IdeaCheck Local ideaCheck: IdeaCheck	**Message to Persisted or Local:** - A new variable is created in the corresponding area. The name of the new variable is derived from the type. - An object flow is added from **message** to the new variable.

Special Case: Adding Persisted Variables

For detailed information see page Persisting Data.

 Message message: IdeaCheck Persisted idea: Idea target formData idea mapFromIdeaCheck	By creating variables as **Persisted**, they are usable in all executions throughout the BPMN model.

 Persisted Enter text employeeName: String id: String idea: Idea personnelNumber: String	To use a persisted variable, click the ▼ icon to open the list of available persisted variables and select the variable you want to use.
	The variable is added to the execution pane.
	ⓘ Once a variable is defined as to be **persisted**, it is available in all executio

ns throughout the BPMN model.

If you drag out the same type again, a second persisted variable is created th at is independent of the first. A consecutive number is appended to the name automatically as variable names

	must be unique.