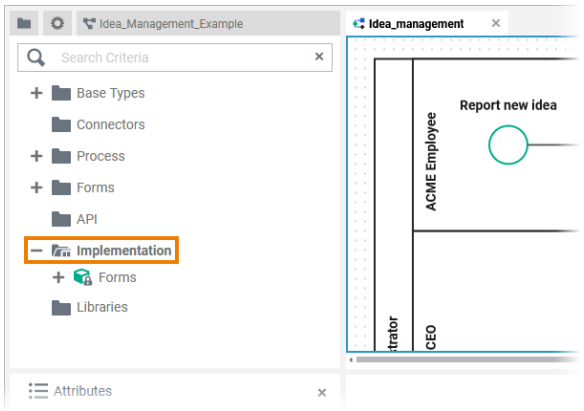
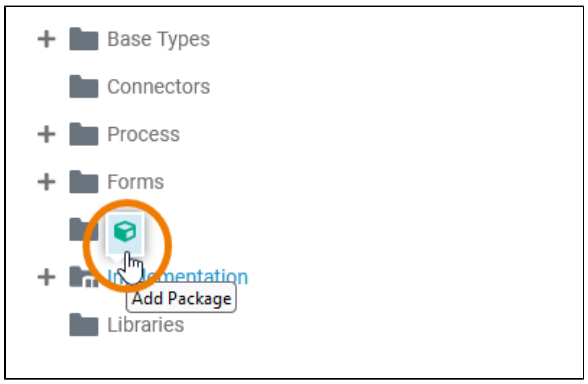
Modeling Data Structures

If you want to create your own data model within the Designer, you need to create a **Service** first. In the **Service** panel resides a folder **Implementation** where you can add your own data model to.



Go to the **Implementation** folder in the service panel of your service.



All elements inside the **Implementation** folder need to be created within packages. Add a new package via the quick action or the context menu. Refer to Service Panel for more information on how to create elements in the service panel.

ExpertAdvice

Apply the same naming c

On this Page:

Related Pages:

- Modeling BPMN
- Modeling Data Mapping
- Modeling Activities
- Using Action Script
- PAS Designer Developer Guide
  - Concepts of Data Modeling

onventionstoallyourmodels.Thismakesreadingamodelmucheasier.RefertoNamingConventionsandCont

ainmentTreeOrganizationintheBuilderUserGuideforanoverviewonpracticeapprovednamingconventions.

Implementation Elements

To model your own data structures, you have the following elements available:

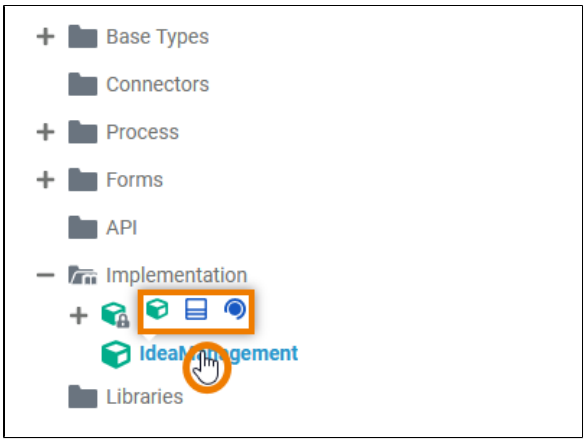
Element	Description	Details
Package	A package is like a directory for the file system. It is used to group executable data model elements. Packages can have any depth of nesting: To structure your work, you can create packages within packages. Also, packages define a sort of namespace to the contained elements. The name of the package is part of the element path, e.g. **Package1.Class** is different from **Package2.Class**.	• Package
Class	A class is an aggregation of properties and operations that describes a complex data type from which objects can be created.	• Class
Property	Properties are data fields that describe the structure of the class.	• Properties
Operation	An operation adds behavior to a class or interface. The behavior describes how to process the data given by the parameters. In the context of the Designer, you can implement operations as mapping, action script or activity.	• Operation
Parameter	Operations can have parameters that define the input and output objects. Operation parameters can be of simple type (Base Types) or of complex type (class or interface).	• Parameter • Adding Parameters to Operations
Operation	Operations can have suboperations with their own parameters.	
Parameter		
...	Suboperations can have suboperations with their own parameters and so on...	
Interface	In contrast to a class, an interface has no properties nor implementations. Interfaces are used to define common operations of multiple classes, and then derive from that interface. Operations of interfaces do not have an implementation but only define the signature (parameters and types).	• Interface
Interface	Interfaces can have sub-interfaces and sub-classes.	
Class		
Operation	Operations and parameters for interfaces are the same as for classes. The difference is that they have no implementation but only define the signature for the dependent classes to derive from.	• Operation
Parameter		• Parameter

Each element of the **Implementation** folder has a context menu and quick actions. The context menu contains options to create new elements to the selected element, and to edit the current element. Via the quick actions, you can access the most used menu items directly with a single click.

> In the Implementation folder, you can undo or redo (after undo) your previous changes using the corresponding functions in the Designer editors or using the corresponding keyboard standard shortcuts (Ctrl+Z/Y).

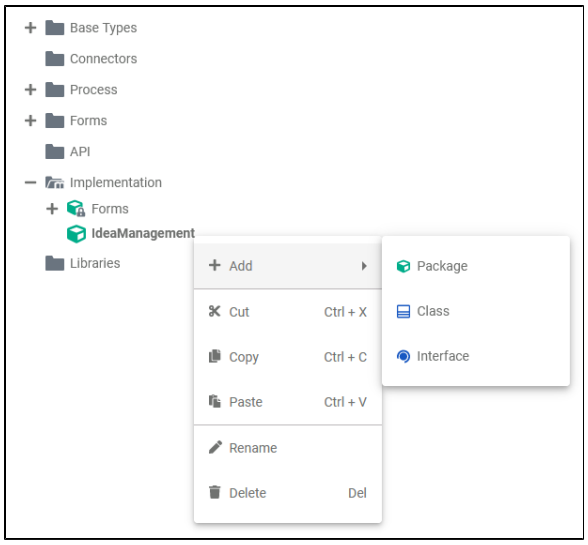
Package

A package is like a directory for the file system. It is used to group executable data model elements. Packages can have any depth of nesting: To structure your work, you can create packages within packages.
Also, packages define a sort of namespace to the contained elements. The name of the package is part of the element path, e.g. **Package1.Class** is different from **Package2.Class**.



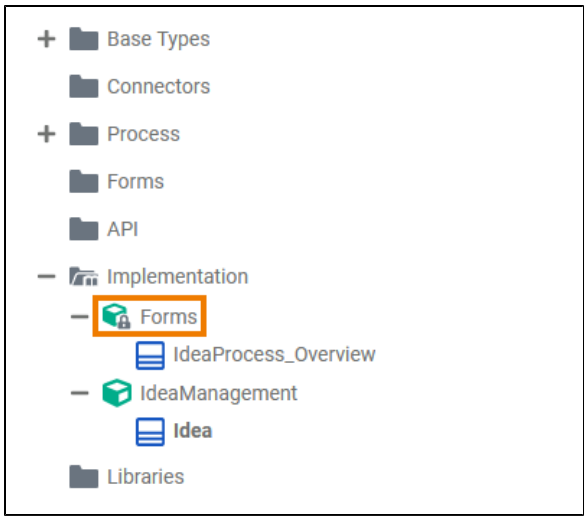
The **quick actions** of a package allow for the creation of packages, classes and interfaces.

Quick Action	Description
	Add another package within the existing package.
	Add a class to the package.
	Add an interface to the package.



The **context menu** of a package allows you to create further elements, to cut, copy and paste the package, to change the name of the package, and to delete it.

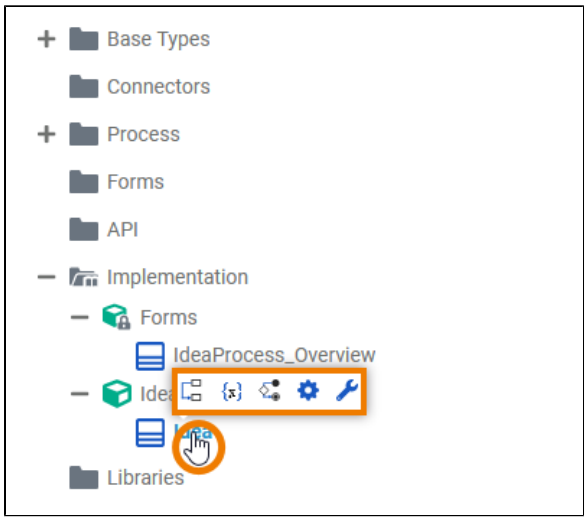
Menu Item	Description
Add Package	Add another package within the existing package.
Add Class	Add a class to the package.
Add Interface	Add an interface to the package.
Cut	Cut the package to paste it elsewhere to the **Implementation** tree.
Copy	Copy the package to paste it elsewhere to the **Implementation** tree.
Paste	Paste the package elsewhere to the data model tree. Available if **Copy** or **Cut** option have been used before.
Rename	Change the package name.
Delete	Delete the package.



Package **Message** is a generated package that contains all classes that are related to the forms you have already created to your **Forms** folder. This package is locked - you cannot change the generated form classes.

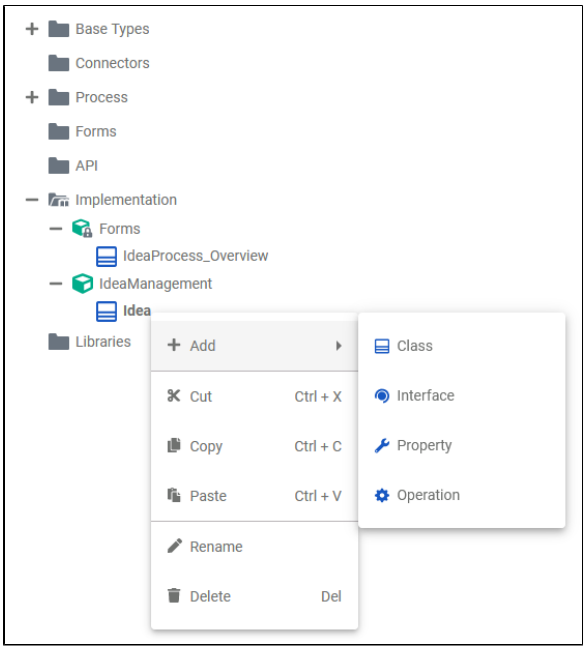
Class

A class is an aggregation of properties and operations that describes a complex data type from which objects can be created.



The **quick actions** of a class allow for the creation of properties as well as operations with different types of implementation.

Quick Action	Description
	Add a mapping operation to the class.
AS	Add an action script operation to the class.
	Add an activity operation to the class.
	Add an operation to the class. The implementation is to be defined later.
	Add a property to the class.

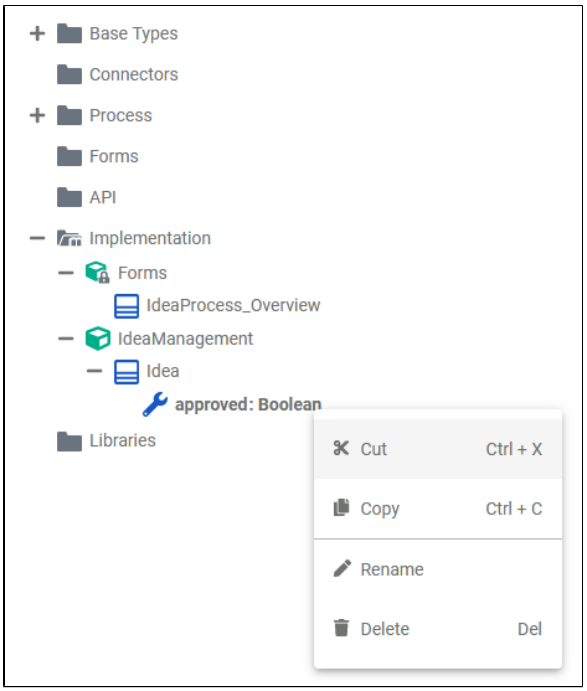


The **context menu** of a class allows you to create further elements, to cut, copy and paste the class, to change the name of the class, and to delete it.

Menu Item	Description
Add Class	Add a sub-class to the class.
Add Interface	Add an interface to the class.
Add Property	Add a property to the class.
Add Operation	Add an operation to the class.
Cut	Cut the class to paste it elsewhere to the **API** or **Implementation** folder.
Copy	Copy the class to paste it elsewhere to the **API** or **Implementation** folder.
Paste	Paste the class elsewhere to the **API** or **Implementation** folder. Available if **Copy** or **Cut** option have been used before.
Rename	Change the name of the class.
Delete	Delete the class.

Property

Properties are data fields that describe the structure of the class.



The **context menu** of a property allows you to cut, copy and paste the property, to change the name of the property and delete it. It is not possible to create further elements below a property.

Menu Item	Description
Cut	Cut the property to paste it elsewhere to the data model tree.
Copy	Copy the property to paste it elsewhere to the data model tree.
Paste	Paste the property elsewhere to the data model tree. Available if **Copy** or **Cut** option have been used before.
Rename	Change the name of the property.
Delete	Delete the property.

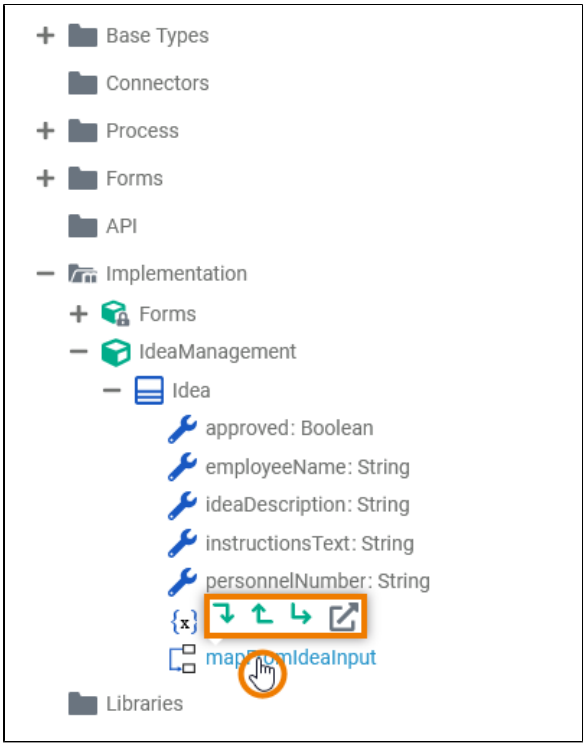
Operation

An operation adds behavior to a class or interface. The behavior describes how to process the data given by the parameters. In the context of the Designer, you can implement operations as:

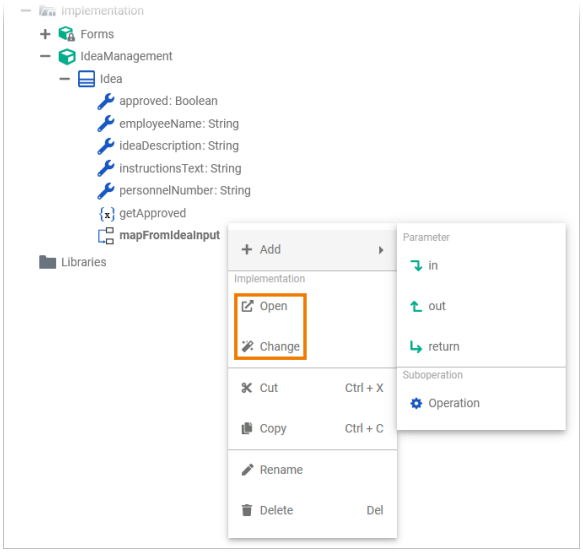
- mapping operation
- activity operation
- action script operation

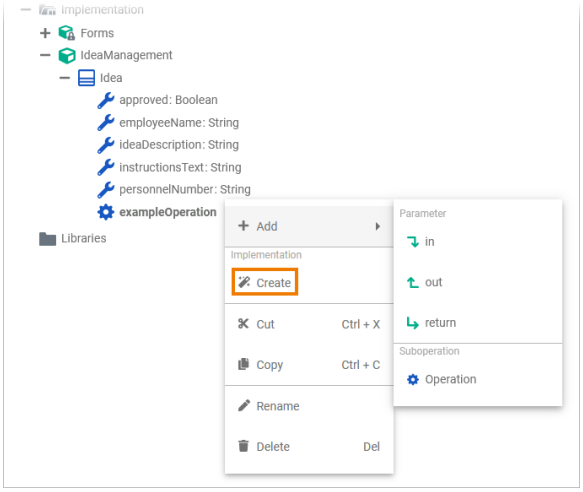
For detailed information on how to implement the different types of operations refer to the the above mentioned pages.

The **quick actions** of an operation allow for the creation of parameters with different directions, and to jump to the implementation of the operation.



Quick Action	Description
	Add an input parameter to the operation.
	Add an output parameter to the operation.
	Add a return parameter to the operation.
	Open the implementation of the operation in a separate tab. If you have not yet selected an implementation, a dialog opens first, which allows you to select the desired implementation. Refer to Create Implementation for more information.





The **context menu** of an operation allows you to create further elements, to select and change the type of implementation of the operation and to open the implementation of the operation. Furthermore you can cut, copy and paste the operation, change the name of the operation, and delete it via this menu.

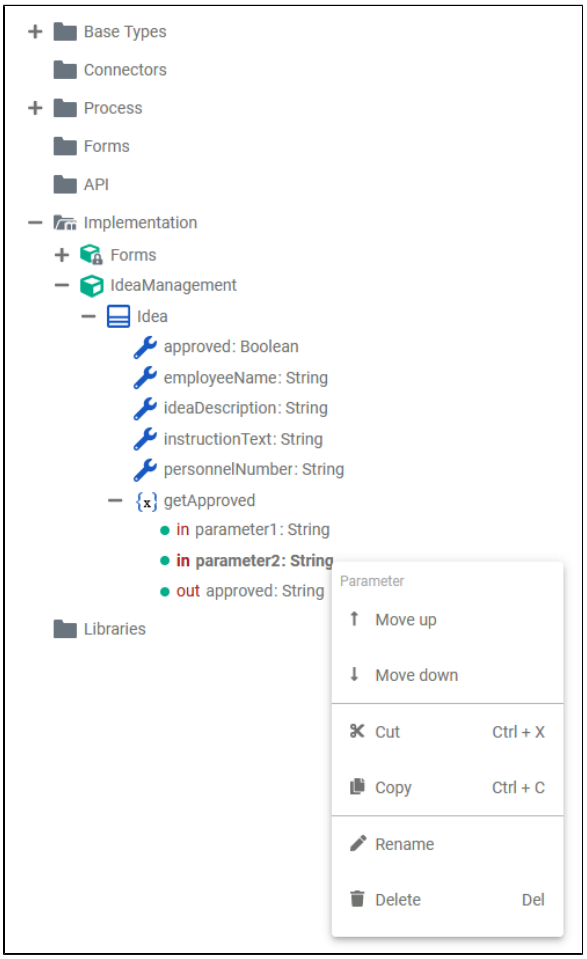
Menu Item	Description
Add Parameter in	Add an input parameter to the operation.
Add Parameter out	Add an output parameter to the operation.
Add Parameter return	Add a return parameter to the operation.
Add Operation (Suboperation)	Add a suboperation to the operation.
Open (Implementation)	Open the implementation of the selected operation in a separate tab. Available if the operation has an implementation, yet.
Change (Implementation)	Change the type of implementation or remove the implementation. Available if the operation has an implementation, yet.

Create (Implementation)	You can choose between three different types of implementation for your class operations: • **Mapping Diagram:** Refer to Modeling Data Mapping for detailed information. • **Action Script:** Refer to Using Action Script for detailed information. • **Activity Diagram:** Refer to Modeling Activities for detailed information. Available if the operation has no implementation, yet.
Cut	Cut the operation to paste it elsewhere to the **API** or **Implementation** folder.
Copy	Copy the operation to paste it elsewhere to the **API** or **Implementation** folder.
Paste	Paste the operation elsewhere to the **API** or **Implementation** folder. Available if **Copy** or **Cut** option have been used before.
Rename	Change the name of the operation.

Delete	Delete the operation.

Parameter

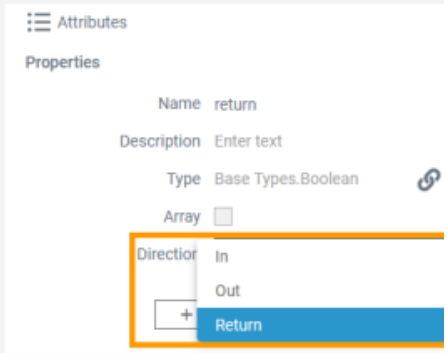
Operations can have parameters that define the input and output objects. Operation parameters can be of simple type (Base Types) or of complex type (class or interface).
Refer to Adding Parameters to Operations for information on how to create parameters.



The **context menu** of a parameter allows you to change th of parameters as well as to change the name of a paramet Furthermore you can cut, copy and paste as well as delete parameter via this menu. It is not possible to create further elements below a parameter.

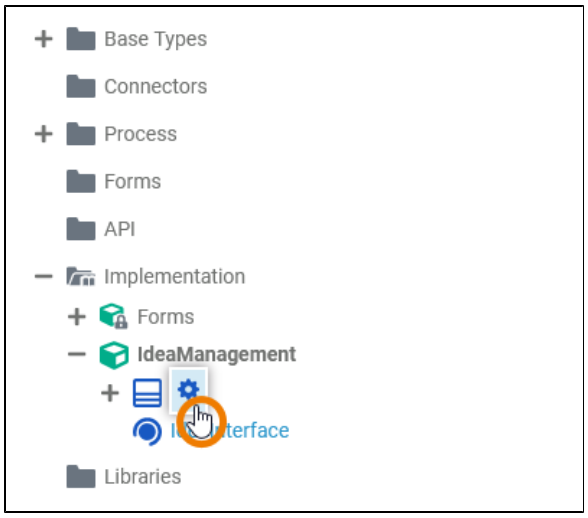
Menu Item	Description
Move up	Change the order of parameters.
Move down	
Cut	Cut the parameter to paste it elsewhere to the **Implementation** folder.
Copy	Copy the parameter to paste it elsewhere to th or **Implementation** folder.
Paste	Paste the parameter elsewhere to the **API** or **I entation** folder. Available if **Copy** or **Cut** optio been used before.
Rename	Change the name of the parameter.
Delete	Delete the parameter.

If you want to change the direction of a parameter, select the parameter and change attribute **Directio** the **Attributes** panel:



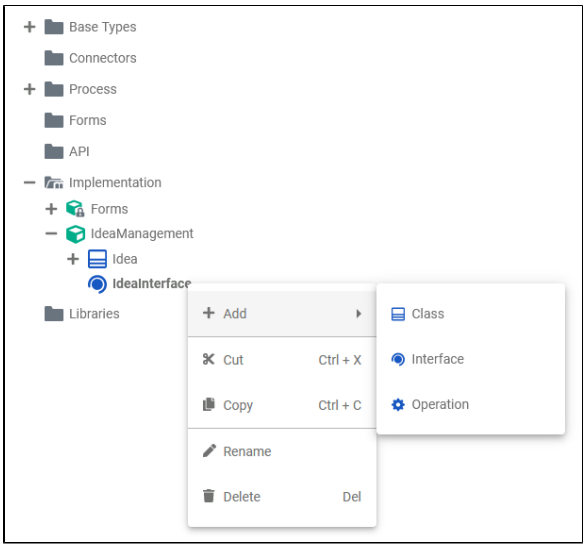
Interface

In contrast to a class, an interface has no properties nor implementations. Interfaces are used to define common operations of multiple classes, and then derive from that interface.
Operations of interfaces do not have an implementation but only define the signature (parameters and types).



The **quick action** of an interface allows for the creation of operations.

Quick Action	Description
	Add an operation to the interface. Operations of interfaces have no implementation, they only describe the signature of the operation.



The **Interface** context menu allows you to create further elements and to change the name of the interface. Furthermore you can cut, copy and paste as well as delete the interface via this menu.

Menu Item	Description
Add Class	Add a class or sub-class to the interface. Classes within interfaces can be nested.
Add Interface	Add another interface to the interface. Interfaces can be nested.
Add Operation	Add an operation to the interface. Operations of interfaces do not have an implementation but only define the signature (parameters and types).
Cut	Cut the interface to paste it elsewhere to the **API** or **Implementation** folder.
Copy	Copy the interface to paste it elsewhere to the **API** or **Implementation** folder.
Paste	Paste the interface elsewhere to the **API** or **Implementation** folder. Available if **Copy** or **Cut** option have been used before.
Rename	Change the name of the interface.
Delete	Delete the interface.