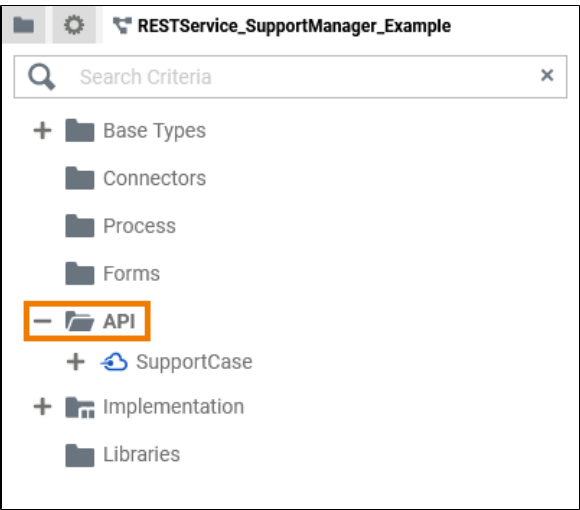


Modeling APIs

In the **Service** panel resides a folder **API** where you can define your own service APIs.



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

On this Page:

- [API Elements](#)
 - [API](#)
 - [Port](#)
 - [Class](#)
 - [Operation](#)
 - [Parameter](#)
 - [Interface](#)

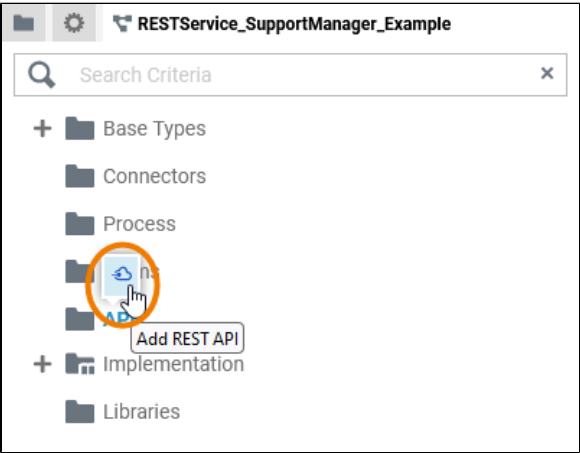
RESTAPI_SupportManager_Example



Click the icon to download a simple example model that shows the implementation of a REST API with **Scheer PAS Designer**.

Related Pages:

- [Modeling BPMN](#)
- [Modeling Data Structures](#)
- [Modeling Data Mapping](#)
- [Modeling Activities](#)
- [Using Action Script](#)
- [PAS Designer Developer Guide](#)
 - [API Implementations](#)



First you need to create the API needed for your service inside the API folder. All other elements will be created within this API. Add a new API via the quick action or the context menu. Refer to [API](#) for more information on how to create a new API in the API folder.

✔ Experience
Apply the same

naming conventions to all your models. This makes reading a model much easier. Refer to [Naming Conventions](#)

API Elements

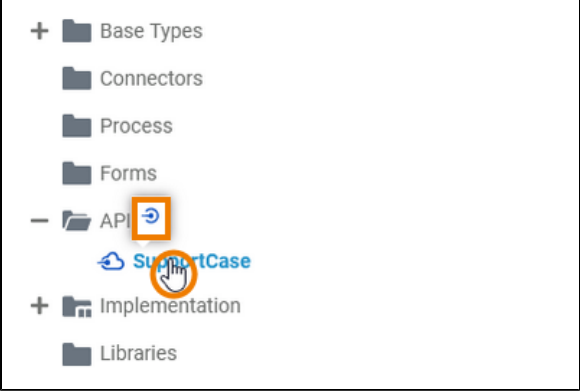



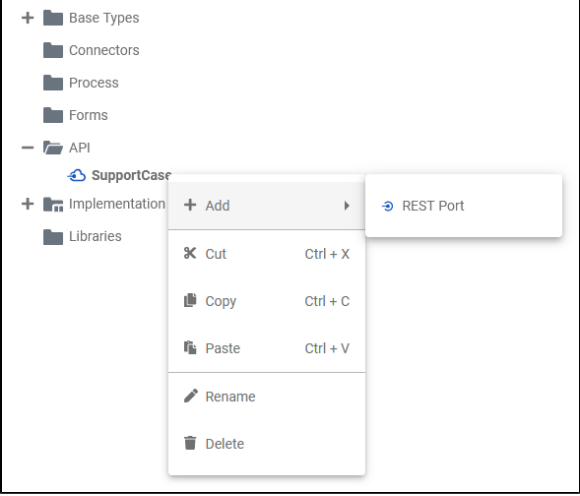
To define a service API, you have the following elements available:

Element	Description	Details
API	An API is the entrance port of your service. It can be used to communicate with the service from the outside. There are different kinds of APIs (e.g. REST API, SOAP API, ...) that describe different communication standards.	<ul style="list-style-type: none"> • API
Port	Ports are elements of the UML modeling standard. A port defines an entrance point to the service and connects the API to an interface or class.	<ul style="list-style-type: none"> • Port
Class	A class is an aggregation of properties and operations that describes a complex data type from which objects can be created.	<ul style="list-style-type: none"> • Class
Class	Classes can have sub-classes.	
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 .	<ul style="list-style-type: none"> • Operation
Parameter	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).	<ul style="list-style-type: none"> • Parameter
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).	<ul style="list-style-type: none"> • 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.	<ul style="list-style-type: none"> • Operation
Parameter		<ul style="list-style-type: none"> • Parameter

Each element of the **API** 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.

API

An API is the entrance port of your service. It can be used to communicate with the service from the outside. There are different kinds of APIs (e.g. REST API, SOAP API, ...) that describe different communication standards.

	<p>The quick action of an API allows for the creation of ports.</p> <table border="1"> <thead> <tr> <th>Quick Action</th><th>Description</th></tr> </thead> <tbody> <tr> <td></td><td>Add a port to the API.</td></tr> </tbody> </table>	Quick Action	Description		Add a port to the API.										
Quick Action	Description														
	Add a port to the API.														
	<p>The context menu of an API allows you to create a port, to cut, copy and paste the API, to change the name of the API, and to delete it.</p> <table border="1"> <thead> <tr> <th>Menu Item</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Add Port</td><td>Add a port to the API</td></tr> <tr> <td>Cut</td><td>Cut the API to paste it elsewhere to the API folder.</td></tr> <tr> <td>Copy</td><td>Copy the API to paste it elsewhere to the API folder.</td></tr> <tr> <td>Paste</td><td>Paste the API elsewhere to the API folder. Available if Copy or Cut option have been used before.</td></tr> <tr> <td>Rename</td><td>Change the API name.</td></tr> <tr> <td>Delete</td><td>Delete the API.</td></tr> </tbody> </table>	Menu Item	Description	Add Port	Add a port to the API	Cut	Cut the API to paste it elsewhere to the API folder.	Copy	Copy the API to paste it elsewhere to the API folder.	Paste	Paste the API elsewhere to the API folder. Available if Copy or Cut option have been used before.	Rename	Change the API name.	Delete	Delete the API.
Menu Item	Description														
Add Port	Add a port to the API														
Cut	Cut the API to paste it elsewhere to the API folder.														
Copy	Copy the API to paste it elsewhere to the API folder.														
Paste	Paste the API elsewhere to the API folder. Available if Copy or Cut option have been used before.														
Rename	Change the API name.														
Delete	Delete the API.														

Port

Ports are elements of the UML modeling standard. A port defines an entrance point to the service and connects the API to an interface or class.

+ Base Types

Connectors

Process

Forms

- API

- SupportCase

SupportAPI

Implementation

Libraries

+ Add

Cut Ctrl + X

Copy Ctrl + C

Rename

Delete Del

Class

Interface

Property

Operation

The **context menu** of a port allows for the creation of classes and interfaces, to change the name of the port, and to delete it.

Menu Item	Description
Add Class	Add a class to the port.
Add Interface	Add an interface to the port.
Rename	Change the port name.
Delete	Delete the port.

Class

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

+ Base Types

Connectors

Process

Forms

- API

- SupportCase

Su

supportcases

Implementation

Libraries





Class

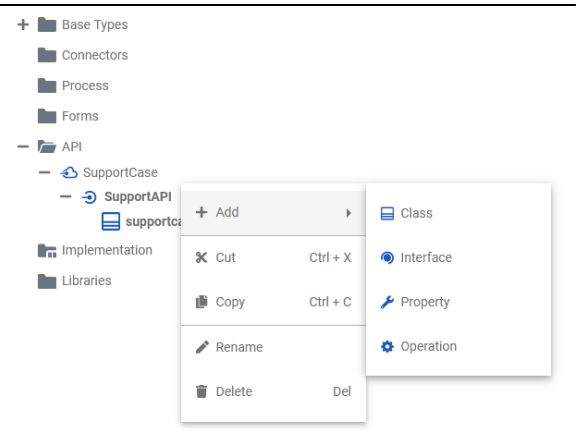
Interface

Property

Operation

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

Quick Action	Description
	Add a mapping operation to the class.
	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.

	<p>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.</p> <table border="1"> <thead> <tr> <th>Menu Item</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Add Class</td><td>Add a sub-class to the class.</td></tr> <tr> <td>Add Interface</td><td>Add an interface to the class.</td></tr> <tr> <td>Add Operation</td><td>Add an operation to the class.</td></tr> <tr> <td>Cut</td><td>Cut the class to paste it elsewhere to the API or Implementation folder.</td></tr> <tr> <td>Copy</td><td>Copy the class to paste it elsewhere to the API or Implementation folder.</td></tr> <tr> <td>Paste</td><td>Paste the class elsewhere to the API or Implementation folder. Available if Copy or Cut option have been used before.</td></tr> <tr> <td>Rename</td><td>Change the name of the class.</td></tr> <tr> <td>Delete</td><td>Delete the class.</td></tr> </tbody> </table>	Menu Item	Description	Add Class	Add a sub-class to the class.	Add Interface	Add an interface 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.
Menu Item	Description																		
Add Class	Add a sub-class to the class.																		
Add Interface	Add an interface 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.																		

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](#).

	<p>The quick actions of an operation allow for the creation of parameters with different directions, and to jump to the implementation of the operation.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

+

Base Types

Connectors

Process

Forms

-

API

-

SupportCase

-

SupportAPI

-

supportcases

+

customer

+

supportcases

+

POST

+

GET





+

+

getMyDate

Implementation

Libraries

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. <div><div><div><div></div><div></div><div></div></div><div>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.</div></div></div>

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: <ul style="list-style-type: none"> • 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).

The screenshot shows a project tree on the left with the following structure:

- Forms
 - API
 - SupportCase
 - SupportAPI
 - supportcases
 - customer
 - supportcase
 - POST
 - GET
 - GET/
 - getByDate
 - in date: DateTime
 - out supportCases: ListOfSupportCases
 - out count: String
 - Implementation
 - Libraries

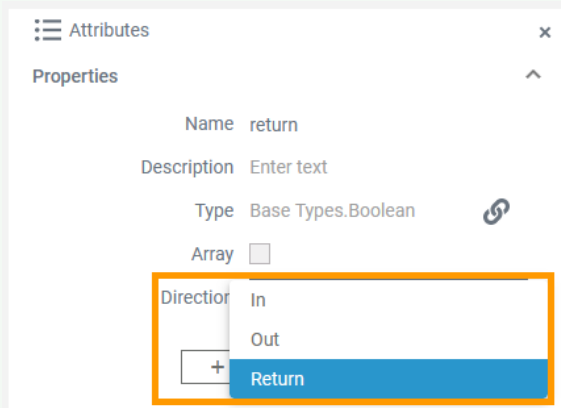
The context menu for the 'getByDate' parameter shows the following options:

- Move up
- Move down
- Cut (Ctrl + X)
- Copy (Ctrl + C)
- Rename
- Delete (Del)

The **context menu** of a parameter allows you to change the order of parameters as well as to change the names of a parameter. Furthermore you can cut, copy and paste a parameter. It is not possible to create further elements below a parameter.

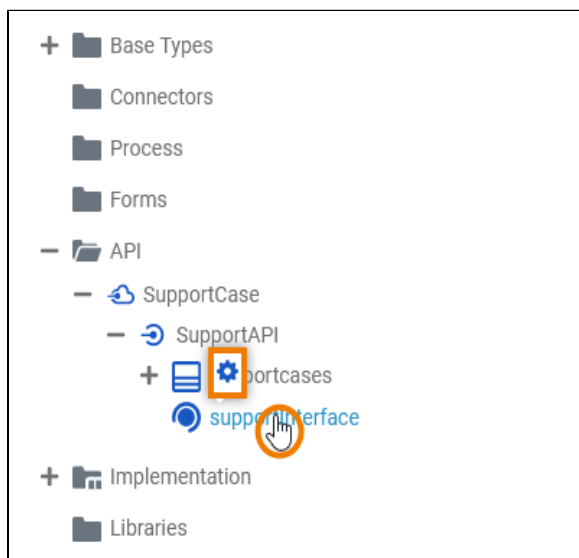
| Menu Item | Description |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Move up | Change the order of parameters. |
| Move down | |
| Cut | Cut the parameter to paste it elsewhere to the API or Implementation folder. |
| Copy | Copy the parameter to paste it elsewhere to the API or Implementation folder. |
| Paste | Paste the parameter elsewhere to the API or Implementation folder. Available if Copy or Cut option have 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 **Direction** in the **Attributes** panel:




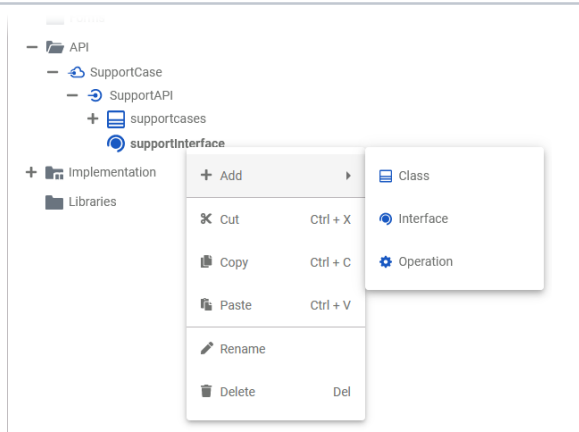
Interface

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. |