

# Adding Operation Calls



You need to perform two steps to implement executonal 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.

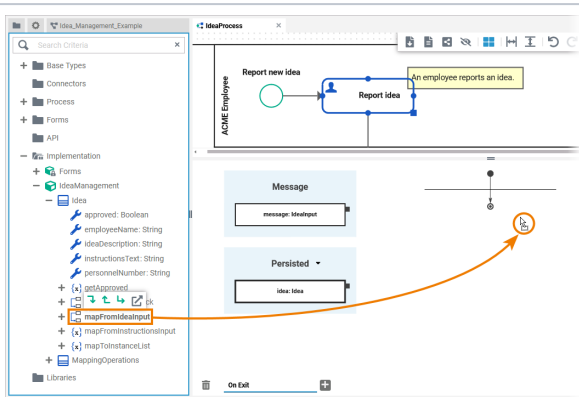
## On this Page:

- [Adding Operations](#)
  - [Pin Highlighting](#)
- [Static And Non-Static Operations](#)

## Related Pages:

- [Implementing Your Process](#)
  - [Modeling Data Structures](#)
  - [Modeling Execution](#)
    - [Adding Variables](#)
    - [Persisting Data](#)
  - [Using Connectors](#)
- [Working With Libraries](#)
- [PAS Designer Developer Guide](#)
  - [Supported BPMN Elements](#)
- [PAS Designer Administration](#)
  - [Adminstrating Libraries](#)

## Adding Operations



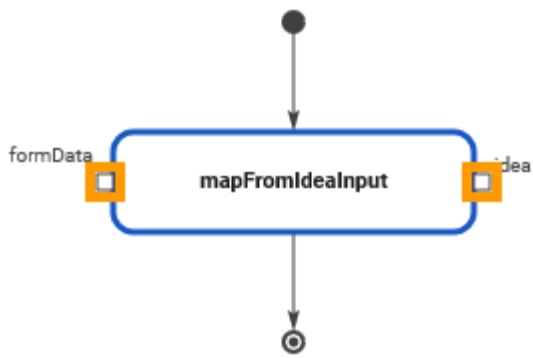
Select the operation you want to use from the service panel.

Drag & drop the operation to the operations flow in the execution pane.



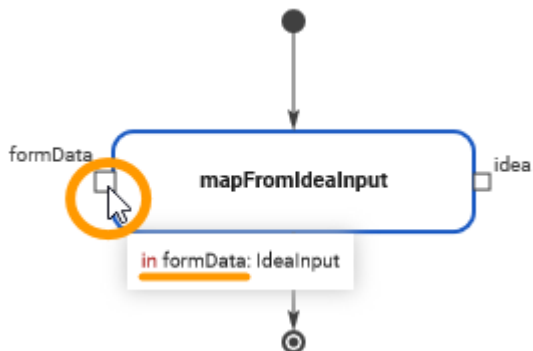
You can do this with any operation: Operations from your

d  
at  
a  
m  
o  
d  
el  
o  
r  
th  
e  
B  
a  
s  
e  
T  
y  
p  
es  
,  
a  
s  
w  
el  
l  
a  
s  
o  
p  
er  
at  
io  
n  
s  
fr  
o  
m  
i  
m  
p  
or  
t  
ed  
li  
b  
r  
a  
ri  
es  
o  
r  
c  
on  
n  
e  
ct  
o  
rs.

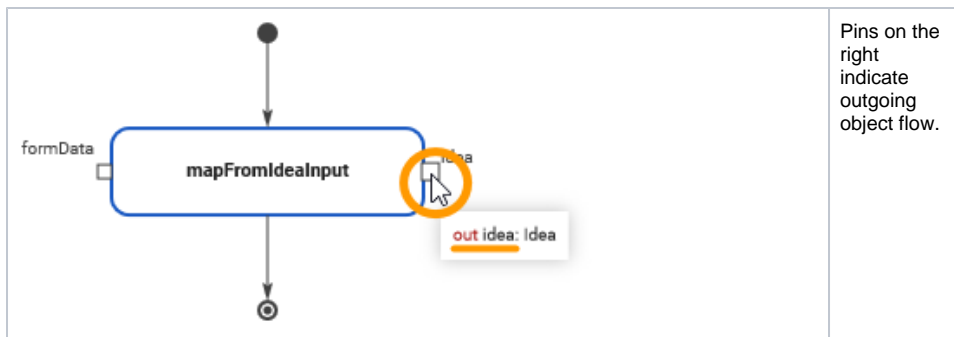


When the operation has been added to the operations flow, the displayed pins indicate needed object flows.

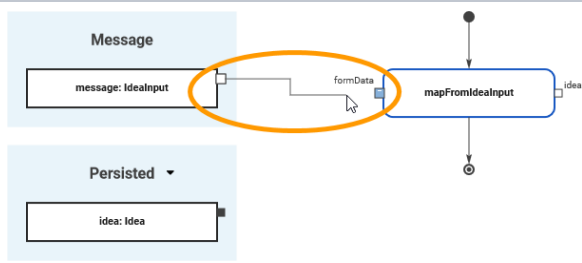
✓ Hover over a pin to see the name and the type of the expected object flow.



Pins on the left of the action node indicate needed incoming object flow.



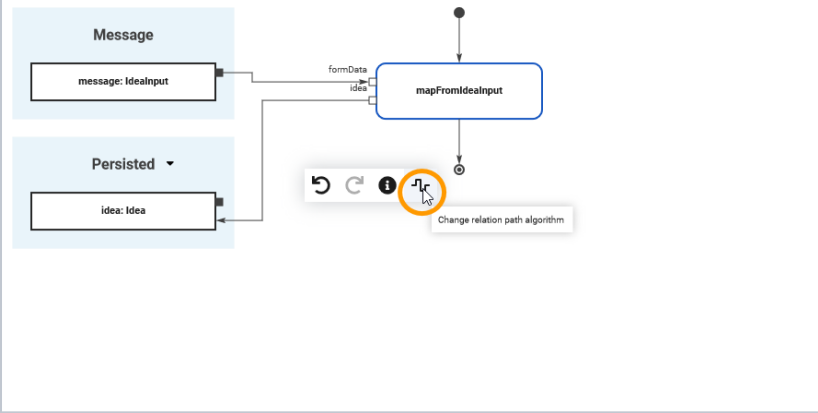
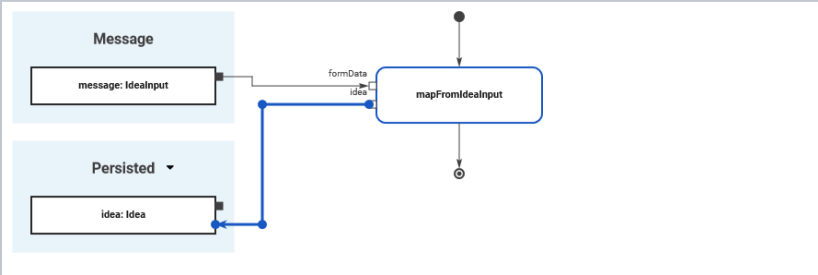
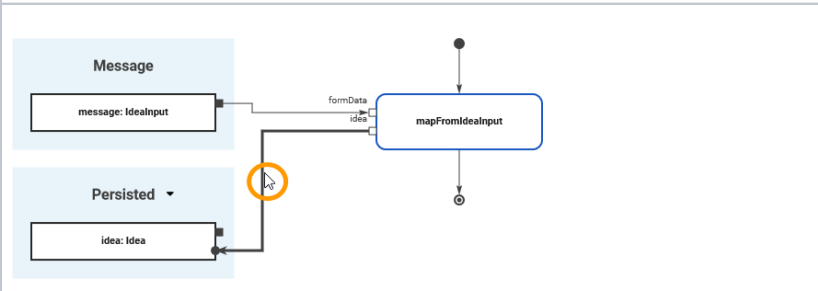
Pins on the right indicate outgoing object flow.



Now you need to connect the pins with the corresponding variables.

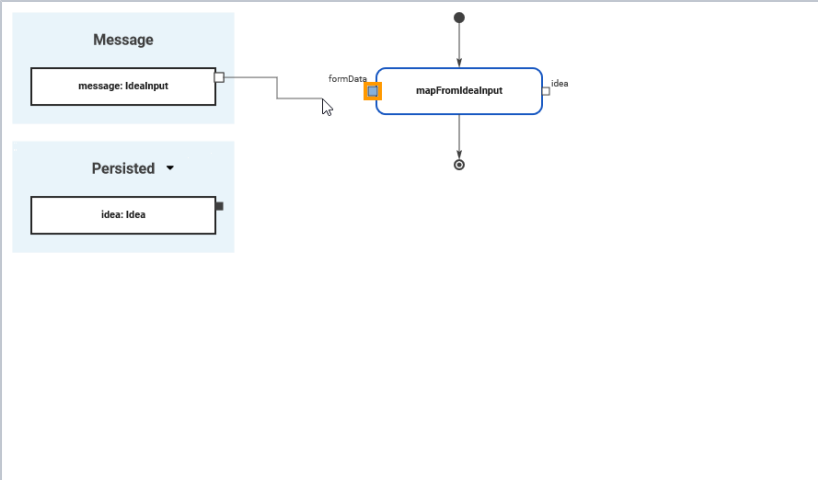
✔ An output pin can be connected to more than one variable.

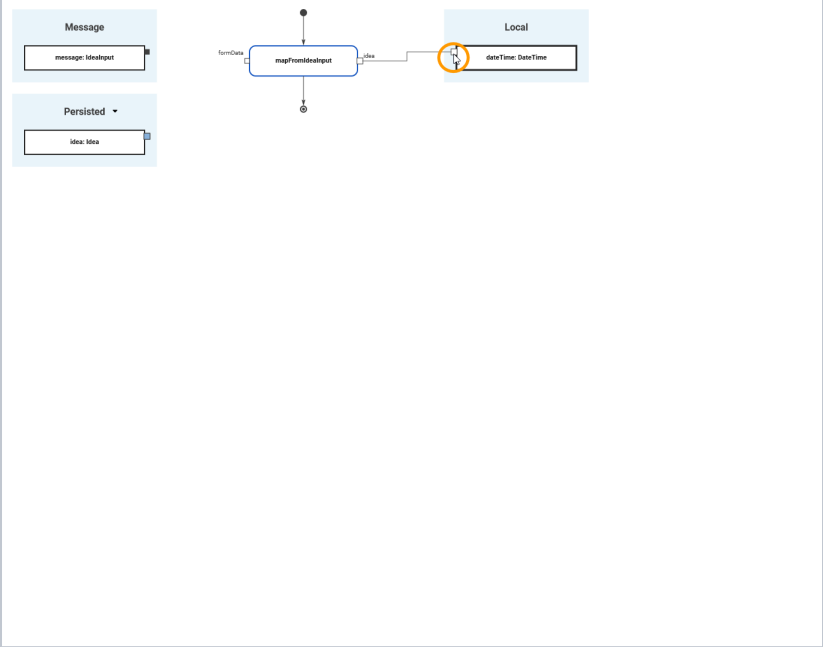
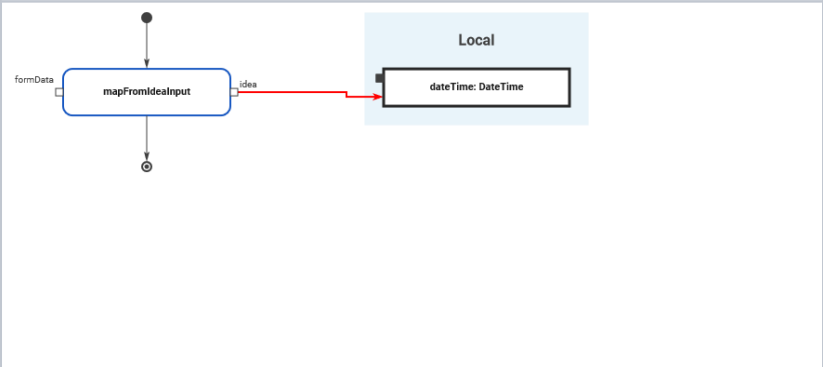
Depending on which data is required, you can use variables from the **Persisted** or **Local** section. Got to [Adding Variables](#) for more information on variables and the difference between persisted and local variables.

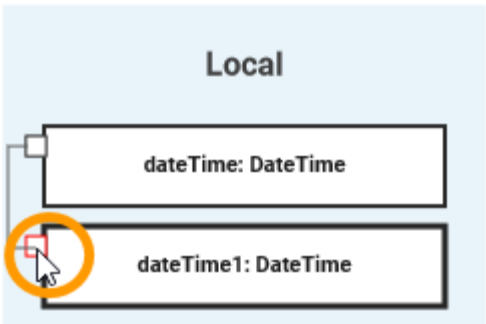
	<p>Object flows on the execution pane are routed automatically.</p> <p>You can use the <a href="#">execution pane context menu</a> to change the algorithm of the relation path.</p>
	<p>You can select an object flow to highlight it in blue - in larger models this helps to track the paths more easily.</p>
	<p>You can also hover over an object flow to highlight it in black.</p>

## Pin Highlighting

While dragging the object flow, the color of the pins changes to assist you with finding matching connection points.

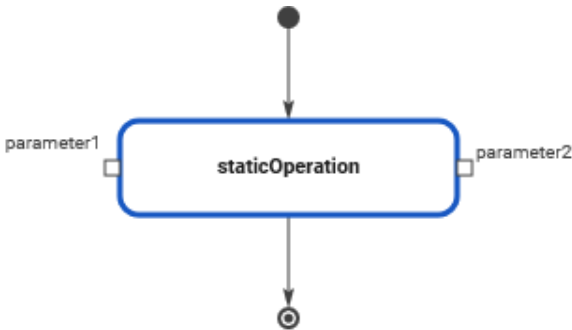
	<p>When you start dragging an object flow, all pins of the same type and multiplicity are marked blue. Blue pins indicate that this pin is a valid target for the object flow.</p>
---	--

	<p>White connection points of variables indicate that you are trying to connect variables that do not have the same type or multiplicity. The connection point is marked in white when you have reached it with the mouse pointer.</p>
	<p>It is possible to draw an object flow to an invalid connection point but the relation will be invalid and displayed in red.</p>

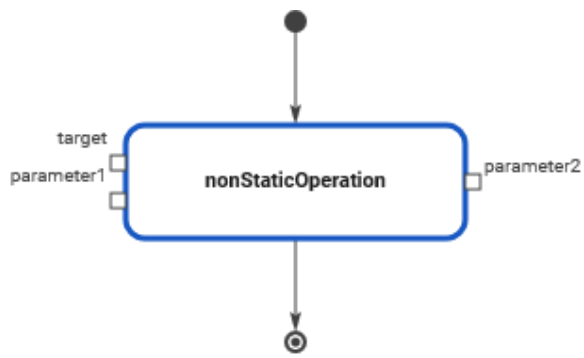
	<p>A pin turns red if an object flow cannot be created at all.</p> <p>You cannot</p> <ul style="list-style-type: none"> <li>• connect a local variable with another local variable</li> <li>• connect a persisted variable with another persisted variable</li> <li>• draw a connection to a pin that already has an incoming object flow.</li> </ul>
---	---

## Static And Non-Static Operations

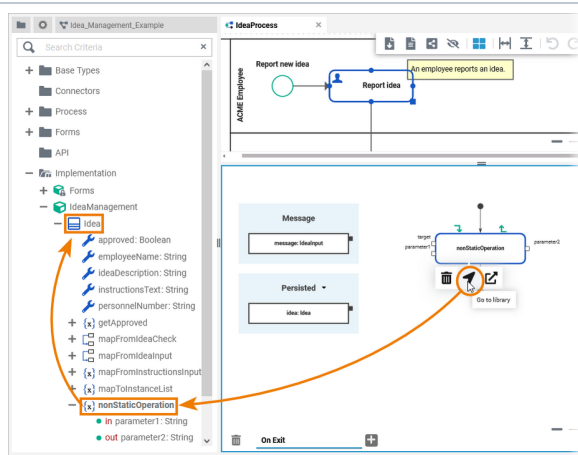
Operations can be **static** or **non-static**.

	<p><b>Static operations</b> are like functions. They have no self context, and thus no target pin - only input, output, and return pins.</p>
---	--





**Non-static operations** are related to a specific type. When added to the execution pane, they provide a **target** pin in addition to the parameters. Here, the user must provide an object of the related type.



To determine the target of a non-static operation, you can:

- **Jump to the service pane I:** The needed target is the class where the operation resides in (see screenshot on the left).
- **Hover over the target pin:** The type of the expected target is displayed on the execution pane.