

## Interfaces [Functional View] [Logical View]

### Docupedia Export

Author:Mukkamala Kiran (XC-AS/EDA2)  
Date:29-Jan-2024 11:23

## Table of Contents

<b>1 Overview</b>	<b>4</b>
<b>2 Big Picture</b>	<b>6</b>
<b>3 Training Material</b>	<b>7</b>
<b>4 Basic structure of example system</b>	<b>8</b>
4.1 Block relations	8
4.2 Activities of subsystem blocks	9
4.2.1 Activities: assigning types to pins and parameters	10
4.3 Adding proxy ports to blocks	12
4.4 Allocating activity parameters to proxy ports	12
4.5 Type classifiers	14
4.6 Defining "complex" interface Types	16
4.6.1 Specifying types	18
<b>5 Elements and Diagrams</b>	<b>20</b>
<b>6 Relevant SysML Terms and Definitions</b>	<b>22</b>
<b>7 Links and further Documentation</b>	<b>28</b>

- Overview
  - Big Picture
  - Training Material
  - Basic structure of example system
    - Block relations
    - Activities of subsystem blocks
      - Activities: assigning types to pins and parameters
    - Adding proxy ports to blocks
    - Allocating activity parameters to proxy ports
    - Type classifiers
    - Defining "complex" interface Types
      - Specifying types
  - Elements and Diagrams
  - Relevant SysML Terms and Definitions
  - Links and further Documentation
-

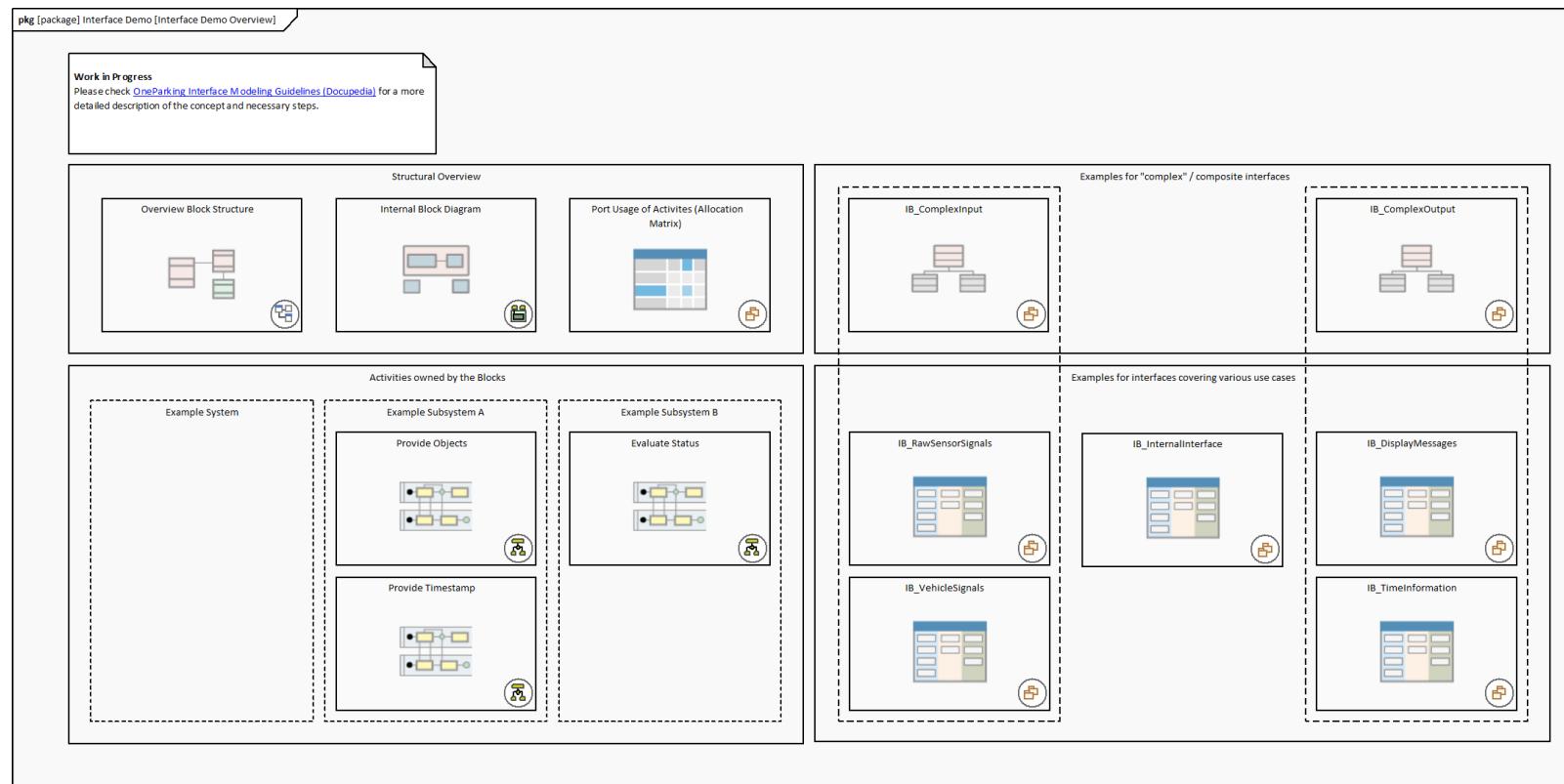
# 1 Overview

In this guideline, we will cover interfaces in both the functional and the logical view.

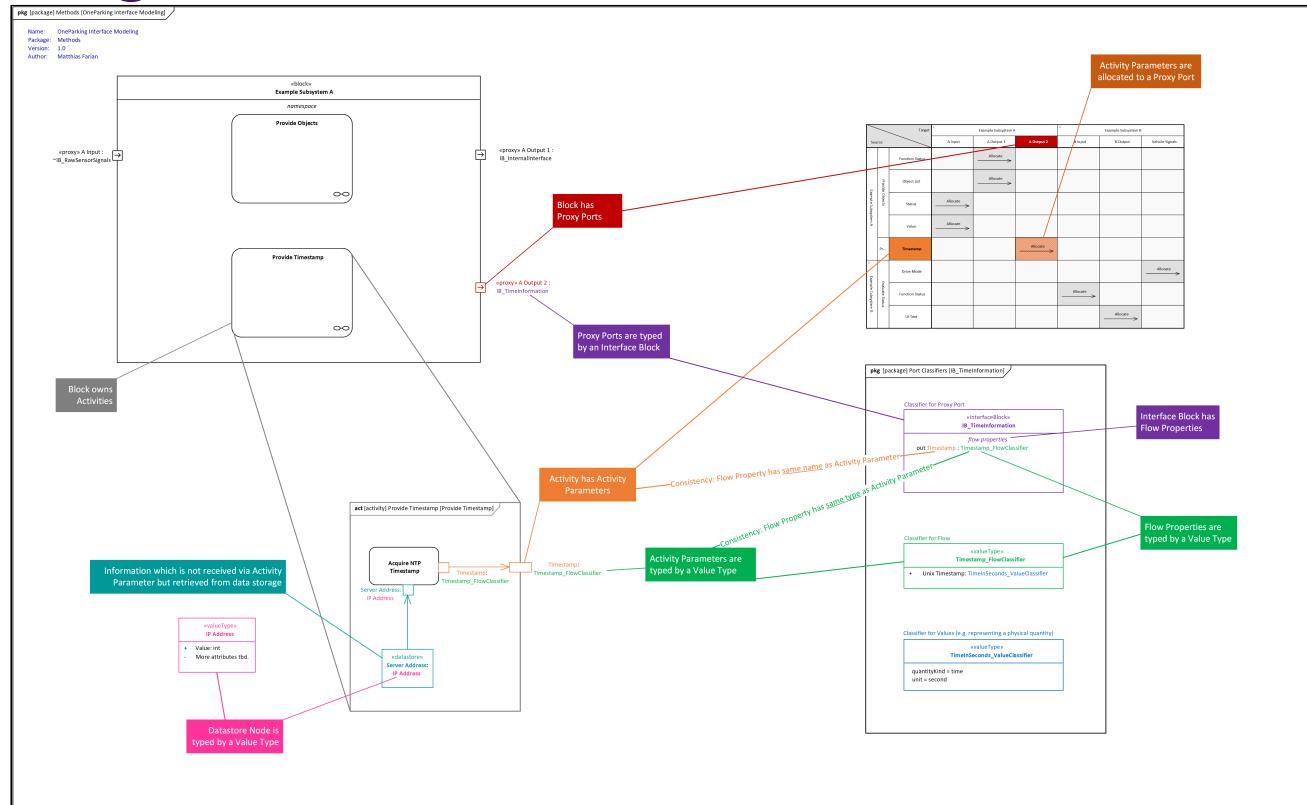
We will take a closer look at modeling

- Inputs and outputs of actions and activities
- Ports of logical blocks
- Allocation of inputs and outputs of activities to ports of logical blocks
- Type definition of
  - Ports
  - Flows
  - Value properties
- Examples for
  - Basic interfaces
  - Complex interfaces with inheritance
  - Lists with variable entries
  - Simple types
  - Custom value types
  - Enumerations

You can find [this example in WebEA](#):



## 2 Big Picture



### Info

**i** This chapter contains a reusable documentation segment ([PageWithExcerpt = Interfaces \[Functional View\] \[Logical View\]](#) | [MultiExcerptName = 1P\\_MBSE\\_Interface\\_Modeling\\_Big\\_Picture](#))

## 3 Training Material



### OneParking Interface Modeling Training 01: Interface Essentials

Matthias Farian (XC-DX/ENG1)  
2023-04-28

 BOSCH

**Slides:** [OneParking\\_Interface\\_Modeling\\_-\\_01\\_Interface\\_Training\\_Essentials.pptx](#)

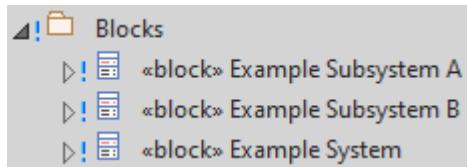
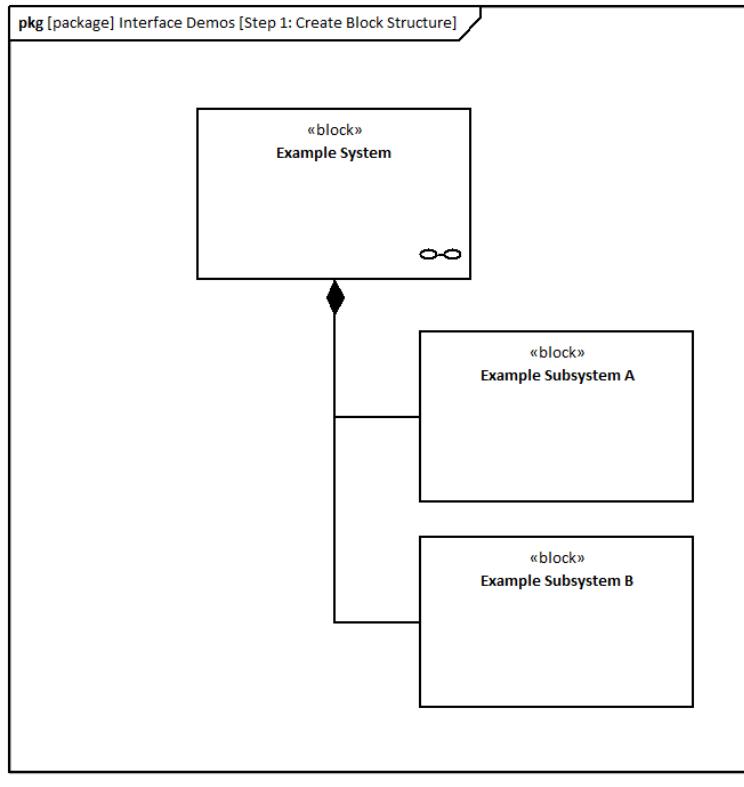
**Enterprise Architect Files:** [OneParking\\_Interface\\_Modeling\\_-\\_01\\_Interface\\_Training\\_Essentials.zip](#)

## 4 Basic structure of example system

This method of interface modeling can be applied to any MBSE level, it might as well be an L3 subsystem consisting of several L4 blocks.

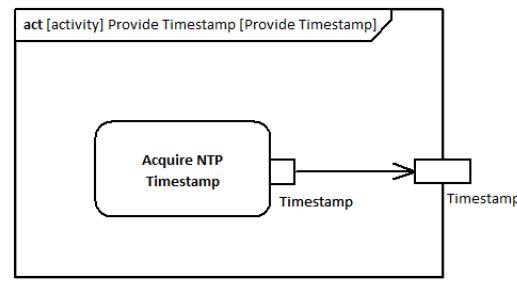
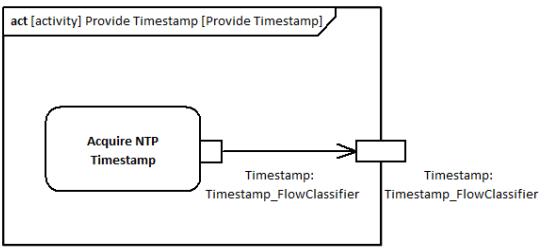
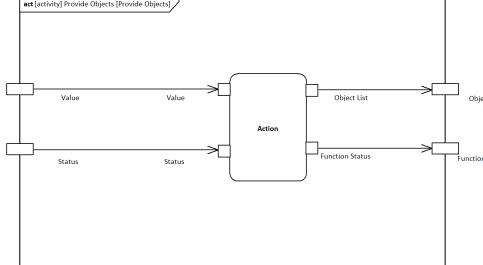
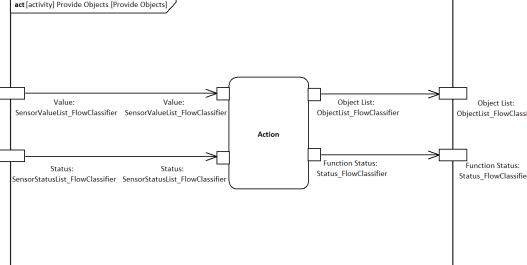
### 4.1 Block relations

This example consists of a simplified system with two exemplary subsystem blocks.

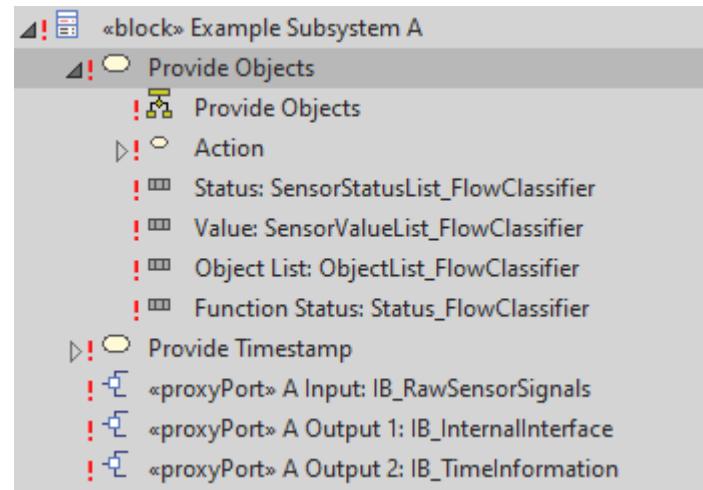


## 4.2 Activities of subsystem blocks

Each block can own multiple activities, with any number of inputs and outputs. Activity parameter nodes act as interfaces for activities.

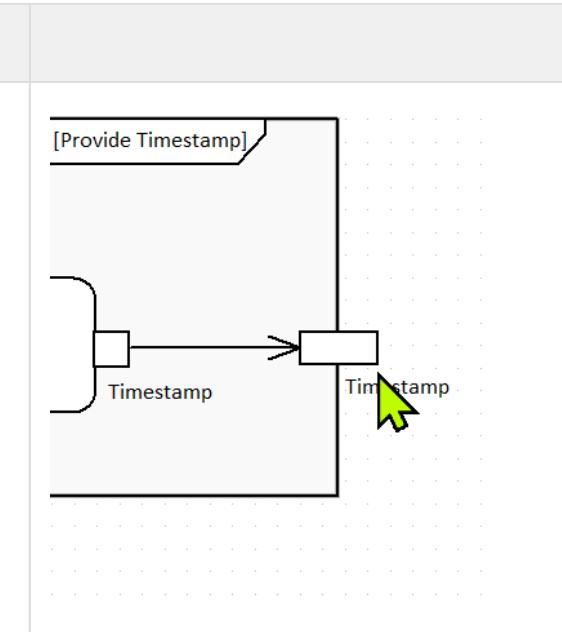
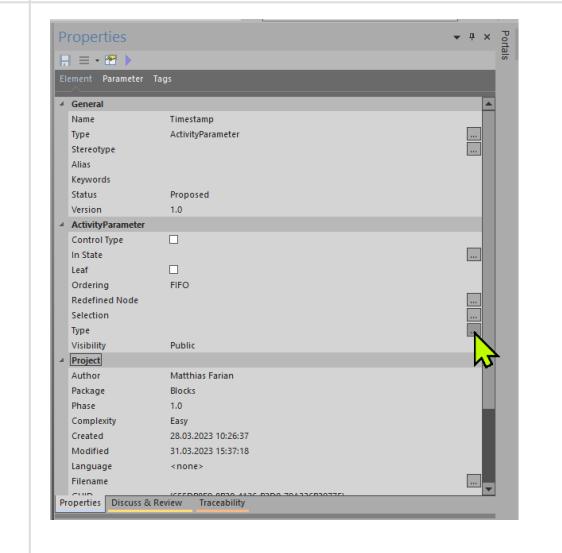
Activity	Step 1: Create Activity Parameters and Action Pins	Step 2: Assign Type Classifier to Pins and Parameter Nodes	
<p>Subsystem A: "Provide Timestamp"</p> <p>Basic Example with just a single output</p>	 <pre> graph TD     subgraph "act [activity] Provide Timestamp [Provide Timestamp]"         direction TB         A[Acquire NTP Timestamp] --&gt; B[ ]         B --&gt; C[Timestamp]     end   </pre>	 <pre> graph TD     subgraph "act [activity] Provide Timestamp [Provide Timestamp]"         direction TB         A[Acquire NTP Timestamp] --&gt; B[ ]         B --&gt; C["Timestamp: Timestamp_FlowClassifier"]         C --&gt; D["Timestamp: Timestamp_FlowClassifier"]     end   </pre>	
<p>Subsystem A: "Provide Objects"</p> <p>Extended example with multiple inputs and outputs, consisting of ordered lists</p>	 <pre> graph TD     subgraph "act [activity] Provide Objects [Provide Objects]"         direction TB         A1[Value] --&gt; B1[ ]         A2[Value] --&gt; B1         A3[Status] --&gt; B2[ ]         A4[Status] --&gt; B2         B1 --&gt; C[Action]         B2 --&gt; C         C --&gt; D[Object List]         C --&gt; E[Function Status]         D --&gt; F[Object List]         E --&gt; F     end   </pre>	 <pre> graph TD     subgraph "act [activity] Provide Objects [Provide Objects]"         direction TB         A1["Value: SensorValueList_FlowClassifier"] --&gt; B1[ ]         A2["Value: SensorValueList_FlowClassifier"] --&gt; B1         A3["Status: SensorStatusList_FlowClassifier"] --&gt; B2[ ]         A4["Status: SensorStatusList_FlowClassifier"] --&gt; B2         B1 --&gt; C[Action]         B2 --&gt; C         C --&gt; D["Object List: ObjectList_FlowClassifier"]         C --&gt; E["Function Status: Status_FlowClassifier"]         D --&gt; F["Object List: ObjectList_FlowClassifier"]         E --&gt; F     end   </pre>	

Activity	Step 1: Create Activity Parameters and Action Pins	Step 2: Assign Type Classifier to Pins and Parameter Nodes	
<p>Subsystem B: "Evaluate Status"</p> <p>Extended example with multiple inputs and a single output consisting</p>	<pre> graph TD     Start1(( )) --&gt; PinFunc1[Function Status]     PinFunc1 --&gt; Action1     Action1 --&gt; UIText1[UI Text]     Start2(( )) --&gt; PinDrive1[Drive Mode]     PinDrive1 --&gt; Action2     Action2 --&gt; ISignal1[Internal Signal]     ISignal1 --&gt; Action1     </pre>	<pre> graph TD     Start1(( )) --&gt; PinFunc1[Function Status: Status_Status_FlowClassifier]     PinFunc1 --&gt; Action1     Action1 --&gt; UIText1[UI Text: String]     Start2(( )) --&gt; PinDrive1[Drive Mode: DriveMode_DriveMode_FlowClassifier]     PinDrive1 --&gt; Action2     Action2 --&gt; ISignal1[Internal Signal: Integer]     ISignal1 --&gt; Action1     </pre>	



## 4.2.1 Activities: assigning types to pins and parameters

Assumption: the relevant type classifiers are already available in the model. If not, please compare chapter on specifying type.

<p>Select activity parameter</p>																																																			
<p>Use the "Type" field in the properties window or press &lt;CTRL&gt;&lt;L&gt;</p>	 <p>Properties</p> <table border="1"><tr><td>Name</td><td>Timestamp</td></tr><tr><td>Type</td><td>ActivityParameter</td></tr><tr><td>Stereotype</td><td></td></tr><tr><td>Alias</td><td></td></tr><tr><td>Keywords</td><td></td></tr><tr><td>Status</td><td>Proposed</td></tr><tr><td>Version</td><td>1.0</td></tr><tr><td colspan="2">ActivityParameter</td></tr><tr><td>Control Type</td><td><input type="checkbox"/></td></tr><tr><td>In State</td><td><input type="checkbox"/></td></tr><tr><td>Leaf</td><td><input type="checkbox"/></td></tr><tr><td>Ordering</td><td>FIFO</td></tr><tr><td>Redefined Node</td><td></td></tr><tr><td>Selection</td><td></td></tr><tr><td>Type</td><td></td></tr><tr><td>Visibility</td><td>Public</td></tr><tr><td colspan="2">Project</td></tr><tr><td>Author</td><td>Matthias Farian</td></tr><tr><td>Package</td><td>Blocks</td></tr><tr><td>Phase</td><td>1.Blocks</td></tr><tr><td>Complexity</td><td>Easy</td></tr><tr><td>Created</td><td>28.03.2023 10:26:37</td></tr><tr><td>Modified</td><td>31.03.2023 15:37:18</td></tr><tr><td>Language</td><td>&lt;none&gt;</td></tr><tr><td>Filename</td><td></td></tr></table>	Name	Timestamp	Type	ActivityParameter	Stereotype		Alias		Keywords		Status	Proposed	Version	1.0	ActivityParameter		Control Type	<input type="checkbox"/>	In State	<input type="checkbox"/>	Leaf	<input type="checkbox"/>	Ordering	FIFO	Redefined Node		Selection		Type		Visibility	Public	Project		Author	Matthias Farian	Package	Blocks	Phase	1.Blocks	Complexity	Easy	Created	28.03.2023 10:26:37	Modified	31.03.2023 15:37:18	Language	<none>	Filename	
Name	Timestamp																																																		
Type	ActivityParameter																																																		
Stereotype																																																			
Alias																																																			
Keywords																																																			
Status	Proposed																																																		
Version	1.0																																																		
ActivityParameter																																																			
Control Type	<input type="checkbox"/>																																																		
In State	<input type="checkbox"/>																																																		
Leaf	<input type="checkbox"/>																																																		
Ordering	FIFO																																																		
Redefined Node																																																			
Selection																																																			
Type																																																			
Visibility	Public																																																		
Project																																																			
Author	Matthias Farian																																																		
Package	Blocks																																																		
Phase	1.Blocks																																																		
Complexity	Easy																																																		
Created	28.03.2023 10:26:37																																																		
Modified	31.03.2023 15:37:18																																																		
Language	<none>																																																		
Filename																																																			

<p>Select type classifier</p>	
<p>Result after assigning types to all action pins and activity parameters</p>	

## 4.3 Adding proxy ports to blocks

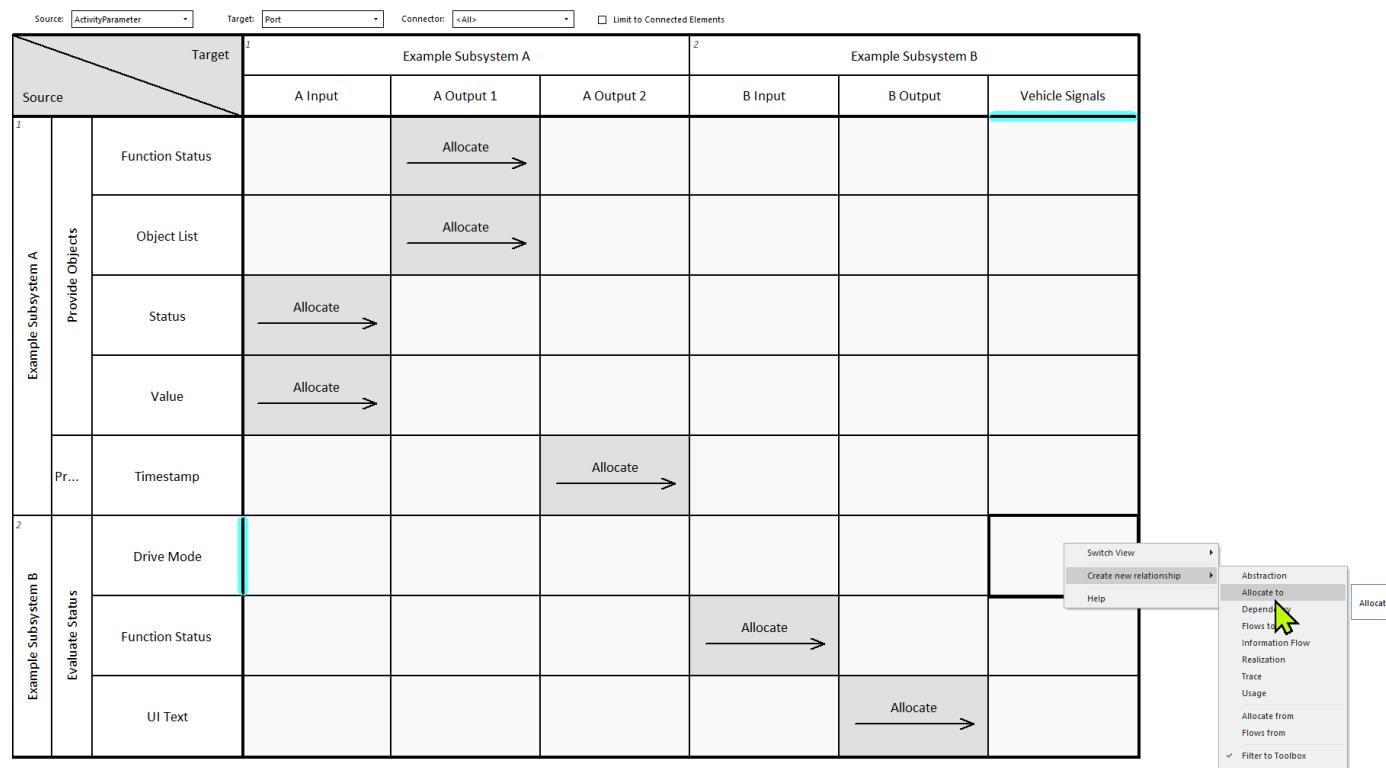
[example will follow soon]

## 4.4 Allocating activity parameters to proxy ports

Pre-requisites:

Proxy ports are typed by interface blocks, owning flow properties. The name of each activity parameter matches the name of a flow property. By allocating the activity parameter to a port, their relation becomes unambiguous.

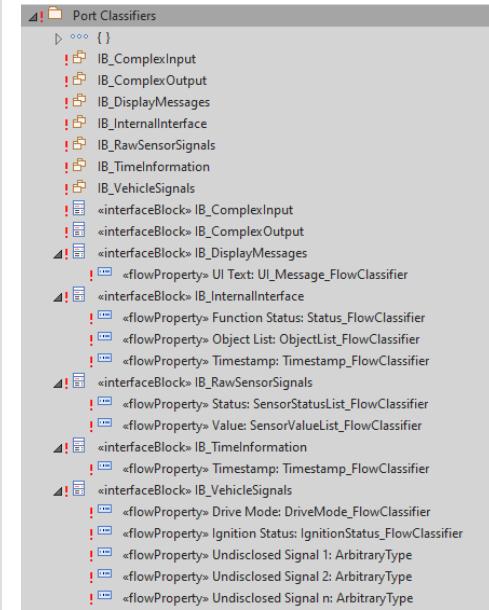
! Make sure to have consistent names for activity parameters and action pins which are connected by object flow connectors.

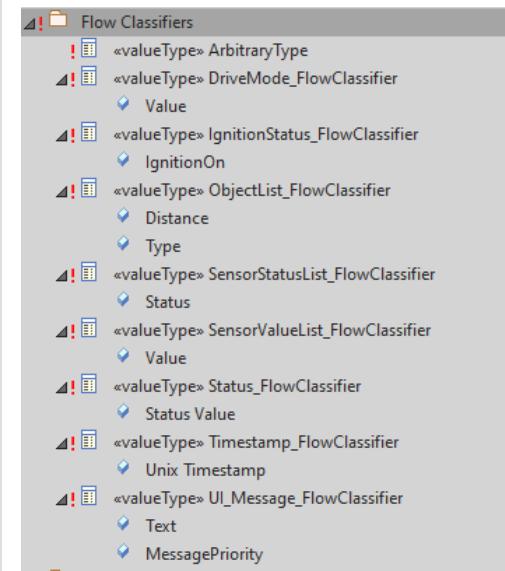
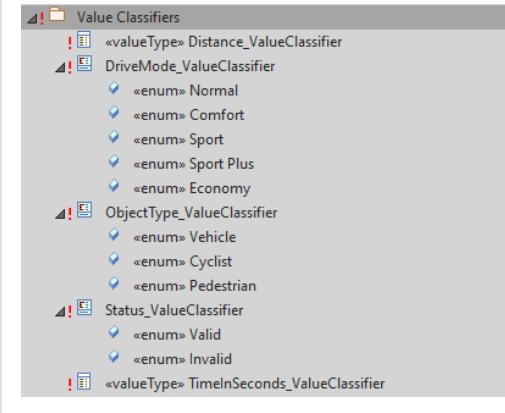


## 4.5 Type classifiers

Interface blocks used as type classifiers for proxy ports

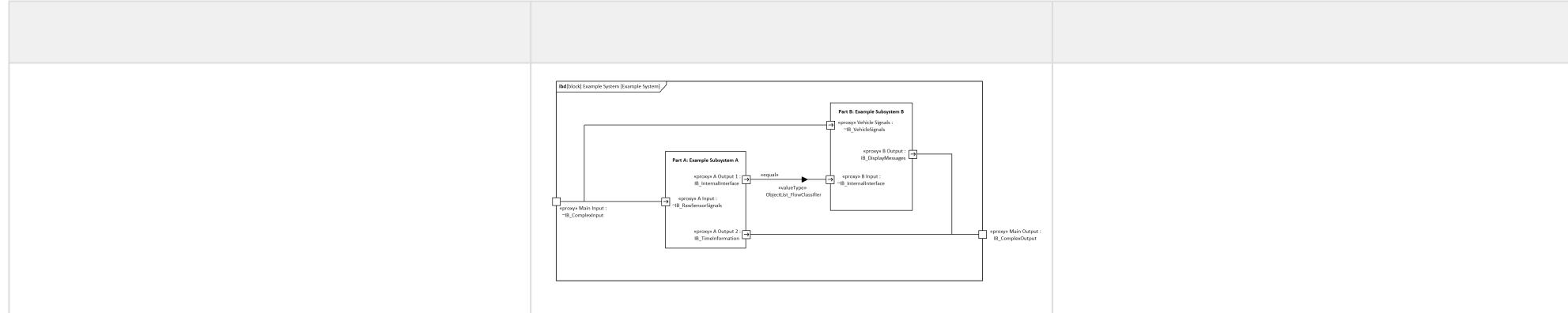
Flow properties represent the individual signals.

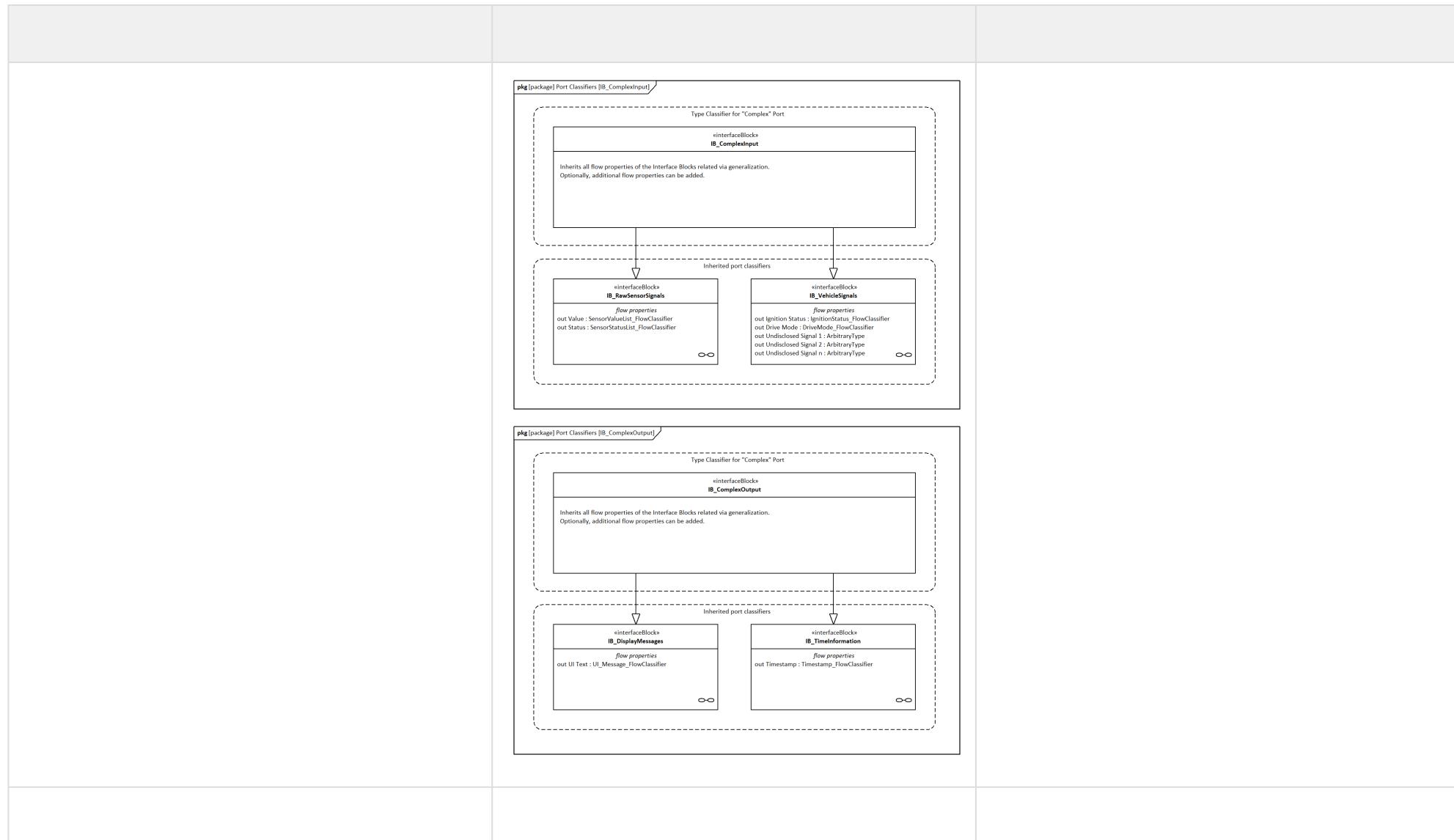


Value types used as type classifiers for flows (individual signals).	
Value types and enumerations used as type classifiers for fine-grained value definition.	
...	

## 4.6 Defining "complex" interface Types

Interface blocks can have generalization relations to other interface blocks. This enables inheriting properties from other interface blocks.

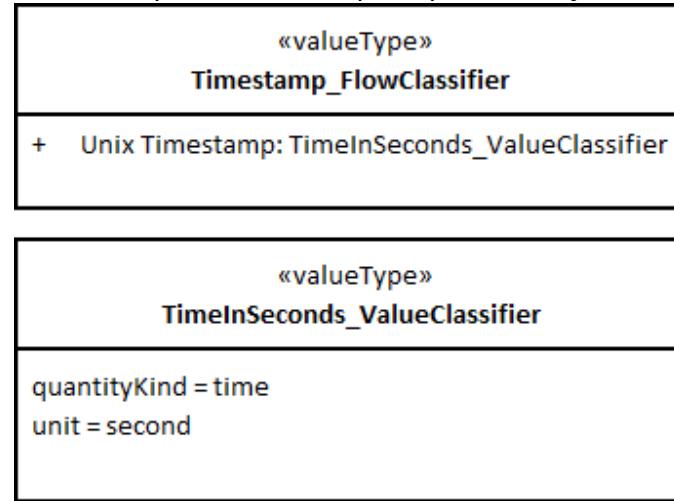




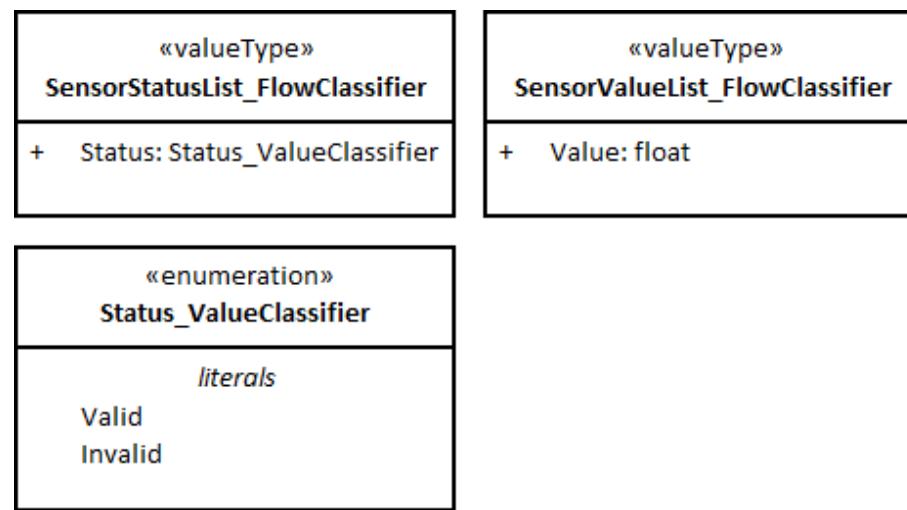
Example for application of complex interface types:

### 4.6.1 Specifying types

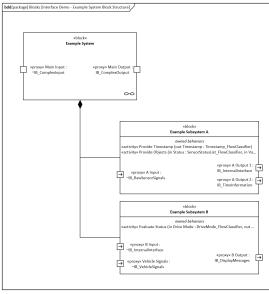
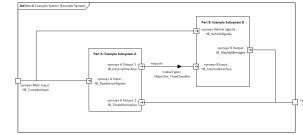
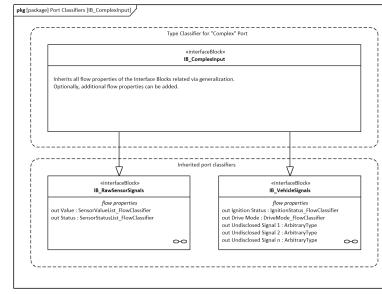
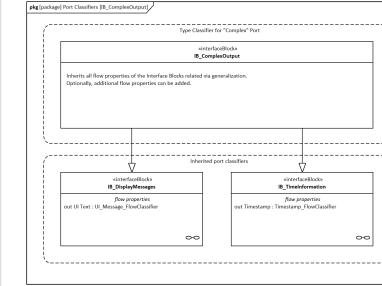
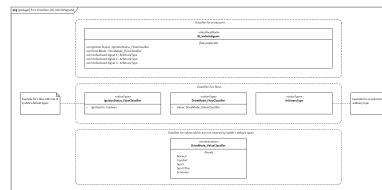
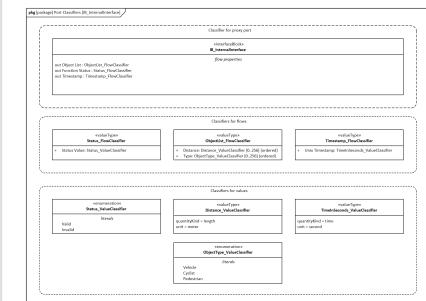
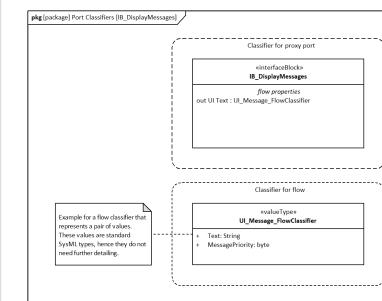
In this example, the timestamp is represented by a Unix timestamp, which itself is a time quantity with unit seconds.



The inputs and outputs of the "Provide Objects" activity are a bit more complicated.



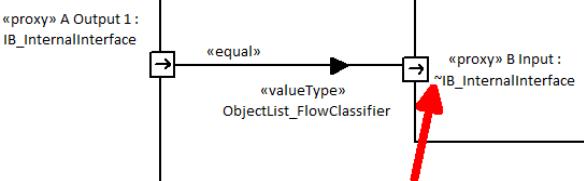
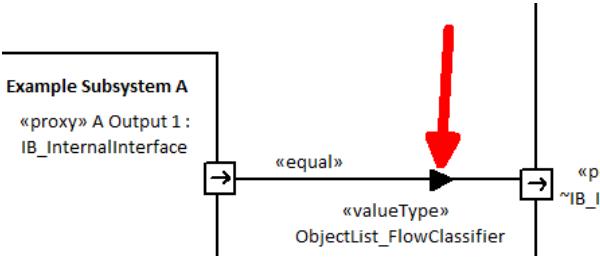
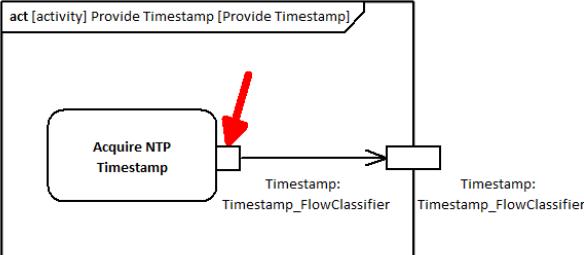
## 5 Elements and Diagrams

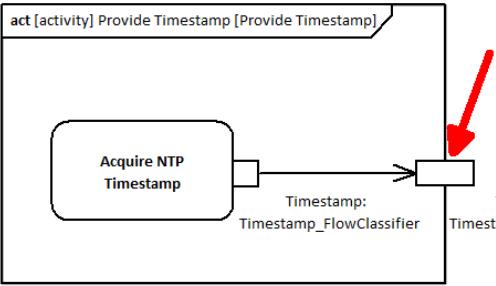
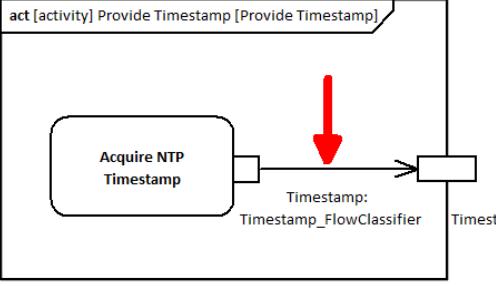
Block structure	Internal block diagram	Interface definition of complex input	Interface definition of internal interface	Interface definition of complex output
 				
				

Block structure	Internal block diagram	Interface definition of complex input	Interface definition of internal interface	Interface definition of complex output

## 6 Relevant SysML Terms and Definitions

Name		Description
Proxy Port		<p>Ports are points at which external entities can connect to and interact with a block. A proxy port acts as proxy for their owning blocks or its internal parts. They define the boundary by specifying which features of the owning block or internal parts are visible through external connectors. (1)</p>
Interface Block	<pre> «interfaceBlock» IB_RawSensorSignals  flow properties out Value : SensorValueList_FlowClassifier[0..16] {unique, ordered} out Status : SensorStatusList_FlowClassifier[0..16] {unique, ordered} </pre>	<p><b>REPLACE SCREENSHOT:</b></p> <ul style="list-style-type: none"> <li>ambiguous multipliers for flows and value properties</li> </ul> <p>Proxy ports are always typed by interface blocks, a specialized kind of block that has no behaviors or internal parts. (1)</p>
Flow Property	<pre> «interfaceBlock» IB_RawSensorSignals  flow properties out Value : SensorValueList_FlowClassifier[0..16] {unique, ordered} out Status : SensorStatusList_FlowClassifier[0..16] {unique, ordered} </pre>	<p><b>REPLACE SCREENSHOT:</b></p> <ul style="list-style-type: none"> <li>ambiguous multipliers for flows and value properties</li> </ul> <p>Flow properties specify the kinds of items that might flow between a block and its environment, whether it is data, material, or energy. The kind of items that flow is specified by typing flow properties.</p> <p>A FlowProperty signifies a single kind of flow element that <i>can</i> flow to/from a block. A flow property's values are either received from or transmitted to another block. (1)</p> <p>Convention in OneParking: Flow Properties shall be modeled with direction "out".</p>

Name		Description
Conjugation		The meaning of the direction is reversed for conjugated ports. (1)
Item Flow	 <p><b>Note:</b> Whereas flow properties specify what “can” flow in or out of a block, item flows specify what “does” flow between blocks and/or parts in a particular usage context. (1)</p>	Item flows specify the things that flow between blocks and/or parts and across associations or connectors.
Action Pin		An Action Pin is used to define the data values passed out of and into an Action. An Input Pin provides values to the Action, whereas an Output Pin contains the results from that Action. (2)

Name		Description
Activity Parameter (Activity Parameter Node)		An Activity Parameter Node accepts input to an Activity or provides output from an Activity. (3)
Object Flow		
Value Type	<div style="border: 1px solid black; padding: 5px;"> <p>«valueType» Timestamp_FlowClassifier</p> <p>+ Unix Timestamp: TimeInSeconds_ValueClassifier</p> </div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>«valueType» TimeInSeconds_ValueClassifier</p> <p>quantityKind = time unit = second</p> </div>	

Name		Description
Value Property	<div style="border: 1px solid black; padding: 10px;"><p>«valueType»</p><p><b>Timestamp_FlowClassifier</b></p><p>+ Unix Timestamp: TimeInSeconds_ValueClassifier</p></div> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"><p>«valueType»</p><p><b>TimeInSeconds_ValueClassifier</b></p><p>quantityKind = time</p><p>unit = second</p></div>	
Enumeration	<div style="border: 1px solid black; padding: 10px;"><p>«enumeration»</p><p><b>Status_ValueClassifier</b></p><p><i>literals</i></p><p>Valid</p><p>Invalid</p></div>	

Name		Description
Quantity Kind	<div style="border: 1px solid black; padding: 10px;"> <p>«valueType»</p> <p><b>Timestamp_FlowClassifier</b></p> <p>+ Unix Timestamp: TimeInSeconds_ValueClassifier</p> </div> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>«valueType»</p> <p><b>TimeInSeconds_ValueClassifier</b></p> <p>quantityKind = time </p> <p>unit = second</p> </div>	
Unit	<div style="border: 1px solid black; padding: 10px;"> <p>«valueType»</p> <p><b>Timestamp_FlowClassifier</b></p> <p>+ Unix Timestamp: TimeInSeconds_ValueClassifier</p> </div> <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>«valueType»</p> <p><b>TimeInSeconds_ValueClassifier</b></p> <p>quantityKind = time </p> <p>unit = second </p> </div>	

**Sources:**

- (1): [OMG Systems Modeling Language, v1.5, Chapter 9 Ports and Flows](#)
- (2): [Spars Systems Enterprise Architect User Guide: Action Pin](#)
- (3): [Spars Systems Enterprise Architect User Guide: Activity Parameter Node](#)



## 7 Links and further Documentation

- [BBM Model-Based Systems Engineering: Interface Concept](#)
- [BBM Model-Based Systems Engineering: Naming Convention Recommendation](#)