


mapAttributes macro

This macro is **deprecated**. Please use the `<<Mapping>>` adapter instead. For details see the chapter [Data Mapping](#).

Syntax	<pre>anInputObject.mapAttributes({anotherInputObject})</pre>	
Semantics	<p>Maps one or more input objects to an output objects. For each attribute that should be mapped, an attribute mapping needs to be defined in a class diagram.</p> <div><p>The way the mapping is done with the mapping macro described in this chapter is deprecated. Refer to section Data Mapping for a description of simple and complex mappings.</p></div> <p>The attribute mapping macro can be used, if you need to map attributes from one object to another object. The attribute mappings are defined directly in a class diagram by drawing dependencies between the attributes of the involved classes. It is possible to create complex rules for the attribute mapping by defining the tagged value mappingRule on target attributes of the output class.</p> <p>Instead of using the attribute mapping macro, you could also use several set statements to reach the same goal. However, it is easier to use the attribute mapping macro, as it reduces the scripting effort. Furthermore, you keep the attribute mappings maintainable, because they are not hidden in the action script, but are well documented in the class diagram.</p>	
Substitutables	<pre>anInputObject, anotherInputObject</pre>	Can be any object having attributes of any type.
Examples	<p>Attributes of object input1 and input2, for which attribute mappings have been defined in a class diagram, are mapped to attributes of object output.</p> <pre>set output = input1.mapAttributes(input2);</pre>	

Example File (Builder projectAdvanced Modeling/Mapping):



```
<your example path>\Advanced Modeling\Mapping\uml\mappingHandler.xml
<your example path>\Advanced Modeling\Mapping\uml\mappingIteration.xml
<your example path>\Advanced Modeling\Mapping\uml\mappingSimple.xml
```

On this Page:

- [UML Example](#)
 - [Attribute Mapping Rules](#)
 - [Example](#)

Related Pages:

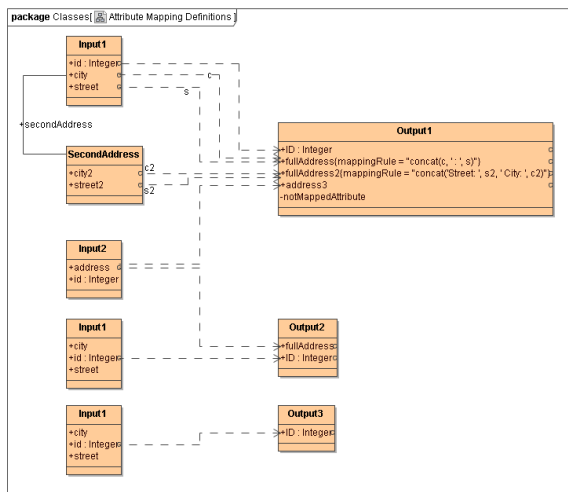
- [Data Mapping](#)

UML Example

The following example shows some simple attribute mappings. Attributes of one or more input objects are mapped to the attributes of an output object. The attribute mappings are defined in a class diagram. In the activity diagram, the input objects have to be defined as input object flow states of the action state that uses the attribute mapping macro. The output object is defined as output object flow state of the same action state.

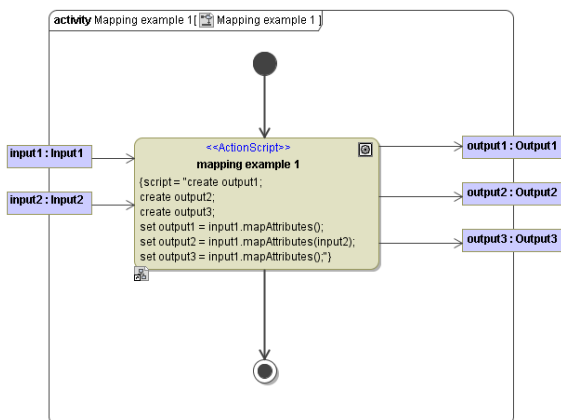
In the following class diagram, the attribute mappings are defined through dependencies between the class attributes.

Figure: Class Diagram with Attribute Mappings



It is possible to map one input attribute to several output attributes. In the example above, **Input2.address** is mapped to **Output1.address3** and **Output2.fullAddress**. If there are more than one dependencies drawn to one single target attribute, the Model Compiler will generate an error, unless a mapping rule for both source attributes is defined. For instance, see target attribute **fullAddress** of class **Output1**. In the example below, some simple attribute mappings are shown.

Figure: Simple Attribute Mapping Operations



When the Model Compiler comes across a call to the `mapAttributes` macro, it will determine the source and destination classes (in the first line of the action script these are the classes **Input1** and **Output1**). The Model Compiler will then search for the class diagram, which contains the attribute mapping definitions for the involved classes. The Bridge also allows you to map attributes of several input and output objects in one action state. For each output object, exactly one call to the `mapAttributes` macro needs to be defined in the action script.

The output object **output1** takes the attributes of the input object **input1** according the defined attribute mapping in the class diagram (see action script statement `set output1 = input1.mapAttributes();`). The output object **output2** takes the attributes of both input objects **input1** and **input2**, so both objects needs be defined as parameters in the mapping statement (see action script statement `set output2 = input1.mapAttributes(input2);`).

For large mappings, it would also be possible to spread the mappings across several class diagrams. The Model Compiler will find the mappings no matter where they are placed.

Attribute Mapping Rules

In the class diagram in figure [Class Diagram with Attribute Mappings](#) above, two mapping rules are defined on the attributes **fullAddress** and **fullAddress2** of class **Output1**. Mapping rules are defined via the tagged value **mappingRule** directly on the target attribute. Complex mapping rules can be applied with this mechanism. You may map from more than one input attribute and process the values before assigning the result to the target attribute.

For instance, the mapping rule `concat('Street: ', s2, ' City: ', c2)` defined in the tagged value **mappingRule** of the attribute **fullAddress2** defines that the target attribute of the output object is assigned a concatenated string composed of two literals ("Street: " and "City: ") and two mapped input attributes. You need to use the name of the dependencies as parameter of the concat operation to reference the correct attributes. In this example, the dependencies **s2** and **c2** are taken as parameters. Verify the class diagram to determine, which attributes are mapped by these dependencies.

Example

Suppose the following example that corresponds to the example above:

Input		Output	
Object.Attribute	Value	Object.Attribute	Value
input1.id	4104	output1.ID	4104
input1.city	Oberwil	output1.fullAddress	Oberwil : Hauptstrasse 3
input1.street	Hauptstrasse 3		
input1.secondAddress.city2	Basel	output2.ID	4104
input1.secondAddress.street2	Lautengartenstrasse 12	output2.fullAddress	Neue Gasse 5
input2.id	4052	output3.ID	4104
input2.address	Neue Gasse 5		

Note, that the value of attribute **output1.fullAddress** was assigned according its mapping rule (compare class diagram in figure [Class Diagram with Attribute Mappings](#)).

It is not possible to link the action specification and the class diagram directly. However, you can place links inside notes in UML diagrams. For a better overview, we recommend that you place a note close to the action state, which has a link that points to the class diagram containing the attribute mappings.

The above examples showed attribute mappings in a standalone context. This makes little sense in real world applications. Of course, attribute mappings can be used at any time and any place in the flow of activities. An example of this would be to map input attributes to column names of a DB table to execute an SQL insert operation.