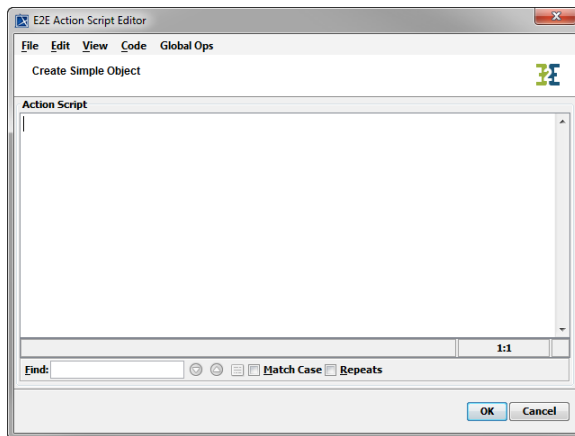


Using the Suggestion Features of the Action Script Editor

The **Action Script Editor** dialog opens.



On this Page:

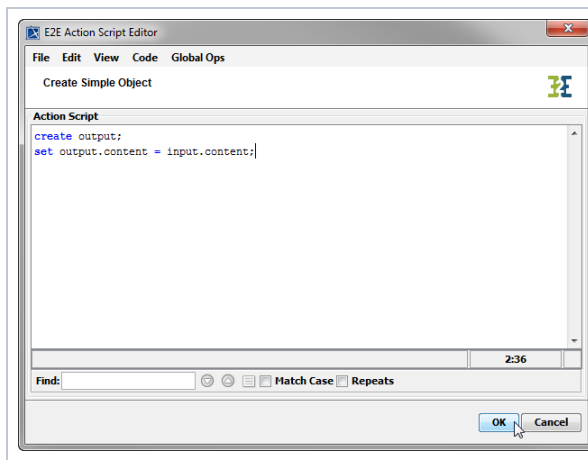
- [Entering a First Action Script Statement](#)
- [Overview on the Suggestion List Features](#)
- [Forcing the Suggestion List](#)

The xUML Action Language is **case sensitive**. Every statement must end with a **semicolon**. Operations, functions, and macros of the action language are displayed in **blue**.

Entering a First Action Script Statement

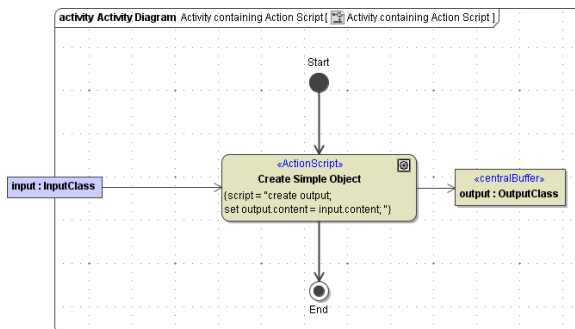
	To create an action script statement, for instance a create assignment statement, start by typing the create keyword. After adding a space, all available object nodes of the action node are listed.
	Add a semicolon to finish the statement.

	<p>Now try the set statement to assign a value to the property of the newly created object. The Action Script Editor automatically adds an equal sign (=).</p> <p>Select the object output from the suggestion list by clicking it. Alternatively, use the up and down arrow keys and confirm a selection by pressing Enter.</p>
	<p>After adding a dot, all properties and operations of the object are listed in the suggestion list.</p> <p>Select the property content by clicking or use the keyboard.</p> <div data-bbox="896 850 1065 1302"> <p>You can use the mouse or the keyboard to select an object, an operation, etc. from the list of suggestions provided by the Action Script Editor. You can also open this list manually by pressing Ctrl together with Space.</p> </div>
	<p>Move the cursor to the end of the line and force the suggestion list by pressing Ctrl + Space.</p>
	<p>Finalize the statement by selecting the property content of object input and adding a semicolon.</p>



The action script for creating an object and assigning a value to a property is complete.

Click **OK** to save the changes and close the editor.



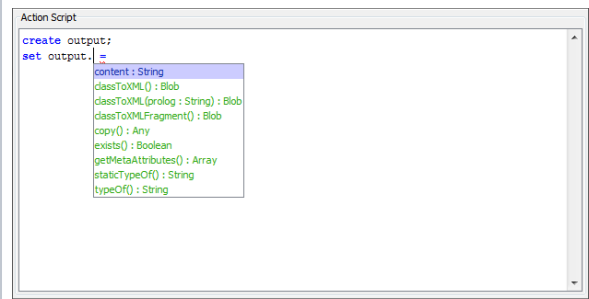
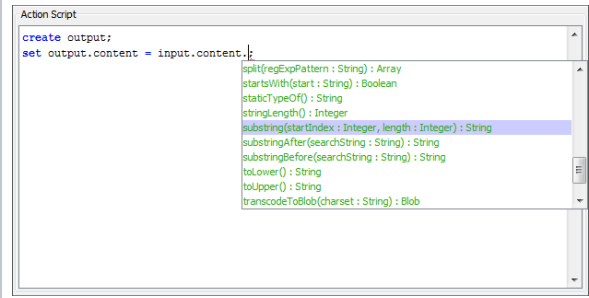
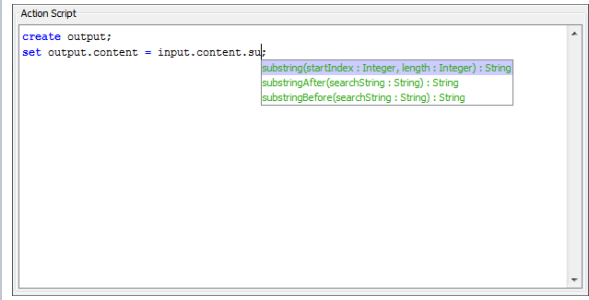
You can recognize an action using an action script by the **<<ActionScript>>** stereotype and the tagged value **script** (see entry {script="..."} in the picture above).

Overview on the Suggestion List Features

As shown in the example above, the Action Script Editor helps you by offering appropriate elements that can be used in the action script statement.



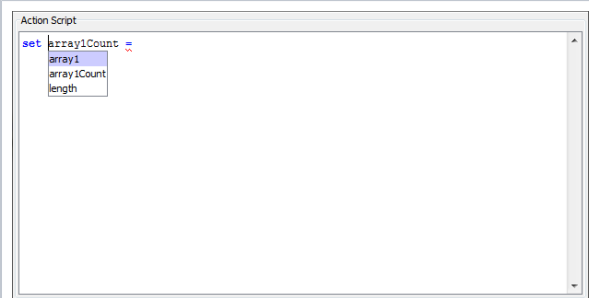
Writing a create statement (e.g. `create output;`, the Action Script Editor offers all output object nodes defined for the action node in the activity diagram.

 <p>The screenshot shows the Action Script Editor with the following code: <code>create output;</code> and <code>set output. =</code>. A suggestion list is open, showing the <code>content</code> property of the <code>output</code> object. The list includes the property name <code>content : String</code> in black, and several methods of the <code>String</code> class in green, such as <code>classToXML() : Blob</code>, <code>classToXML(prolog : String) : Blob</code>, <code>classToXMLFragment() : Blob</code>, <code>copy() : Any</code>, <code>exists() : Boolean</code>, <code>getMetaAttributes() : Array</code>, <code>staticTypeOf() : String</code>, and <code>typeOf() : String</code>.</p>	<p>Referencing an object or an object property (e.g. <code>output</code> or <code>output.content</code>), the Action Script Editor offers all properties and operations defined for the class when the cursor is immediately after the dot following the object name.</p> <p>The Action Script Editor colors object properties in black, attributes of a complex type in blue, and operations in green.</p>
 <p>The screenshot shows the Action Script Editor with the code <code>set output.content = input.content.</code>. A suggestion list is open, showing the <code>split</code> operation of the <code>String</code> class. The list includes the operation name <code>split(regexPattern : String) : Array</code> in black, and several other methods of the <code>String</code> class in green, such as <code>startsWith(start : String) : Boolean</code>, <code>staticTypeOf() : String</code>, <code>stringLength() : Integer</code>, <code>substring(startIndex : Integer, length : Integer) : String</code>, <code>substringAfter(searchString : String) : String</code>, <code>substringBefore(searchString : String) : String</code>, <code>toLowerCase() : String</code>, <code>toUpperCase() : String</code>, and <code>transcodeToBlob(charset : String) : Blob</code>.</p>	<p>Referencing the operations of an object, the Action Script Editor offers each operation with its corresponding parameters.</p>
 <p>The screenshot shows the Action Script Editor with the code <code>set output.content = input.content.sub</code>. A suggestion list is open, showing the <code>substring</code> operation of the <code>String</code> class. The list is filtered to show only the <code>substring</code> operation and its variants: <code>substring(startIndex : Integer, length : Integer) : String</code>, <code>substringAfter(searchString : String) : String</code>, <code>substringBefore(searchString : String) : String</code>, and <code>substringBefore(searchString : String) : String</code>.</p>	<p>The suggestion list can be filtered by starting to type the initial letters of the operation. The filtering is case insensitive.</p>

Forcing the Suggestion List

Forcing the suggestion list by pressing **Ctrl + Space**, the Action Script Editor offers a filtered list dependent on the position of the cursor within the object node name. The Action Script Editor offers only objects, properties and operations that match the filter.

For example, in an activity diagram with object nodes: `array1`, `array1Count`, and `length`:

 <p>The screenshot shows the Action Script Editor with the code <code>set array1Count =</code>. A suggestion list is open, showing the <code>array1Count</code> object node. The list includes the object name <code>array1Count</code> in black, and the <code>length</code> property in blue.</p>	<p>If the cursor is placed before the string "array1Count", the filter is empty and the Action Script Editor offers all 3 object nodes.</p>
--	---

 <p>The screenshot shows the Action Script editor with the code <code>set array1Count =</code>. A suggestion list is open, showing <code>array1</code> and <code>array1Count</code>.</p>	<p>If the cursor is after "arr", the filter contains "arr" and the Action Script editor offers only a <code>rray1</code> and <code>array1Count</code>.</p>
 <p>The screenshot shows the Action Script editor with the code <code>set array1Count =</code>. A suggestion list is open, showing <code>array1Count</code>.</p>	<p>If the cursor is after "array1C", the filter contains "array1C" and the Action Script editor offers only a <code>rray1Count</code>.</p>
 <p>The screenshot shows the Action Script editor with the code <code>set array1Count = array1.</code>. A suggestion list is open, showing various methods and properties of the <code>Array</code> class, such as <code>classToXML()</code>, <code>concatArrays()</code>, <code>copy()</code>, <code>count()</code>, <code>exists()</code>, <code>getMetaAttributes()</code>, <code>staticTypeOf()</code>, and <code>typeof()</code>.</p>	<p>Referencing an array object node (e.g. <code>array1</code>), the Action Script Editor only offers operations that are relevant for arrays.</p>
 <p>The screenshot shows the Action Script editor with the code <code>set array1Count = array1.count();</code> and <code>set length = array1[0].</code>. A suggestion list is open, showing <code>length</code> and <code>length : Integer</code>.</p>	<p>Referencing an array element object (e.g. <code>array1[0]</code>), the Action Script Editor offers all attributes defined for the array element class.</p>
 <p>The screenshot shows the Action Script editor with the code <code>set sum =</code>. A suggestion list is open, showing <code>usePriceCalculator</code> and other methods like <code>createUniqueID()</code>, <code>currentDateTime()</code>, <code>currentLocalDateTime()</code>, <code>currentTimeTicks()</code>, <code>getCompositeHost()</code>, and <code>getCompositeName()</code>.</p>	<p>Static operations of classes can be made available by use-dependencies. These use-dependencies are listed in pink in the suggestion list.</p>

```
Action Script
set sum = usePriceCalculator(
    calculatePrice(price : Float, exchangeRate : Float) : Float
```

After selecting the use-dependency, add a colon to get a suggestion list of all operations that are available via this dependency.