Local Variables

It is possible to define variables that are local within an action script. These local variables are only known within the current action script. They are declared by

```
local nameOfLocalVariable = <expression>;
```

The type of the local variable is derived by the type of expression. Expressions are object accessors, operation calls, boolean expressions, literals, etc.

This enables the compiler to apply static type checking without requiring the modeller to put the static types into the action script. However, sometimes it is necessary to create complex local types.
This is done by statements like

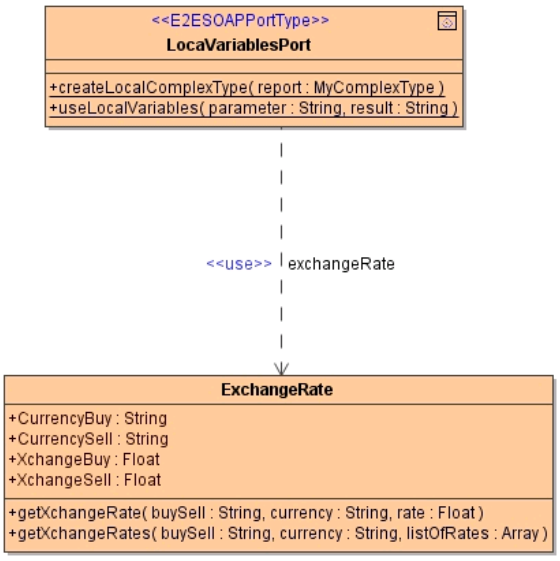
```
create local nameOfLocalVariable using dependencyName;
```

This is similar to an ordinary create statement. The main difference are the keywords **local** and **using**. **Local** means, that the created object is visible only in the current action script. The keyword **using** is used to help the compiler finding the type of the local variable. Since the local variable is not given in the activity diagram as an object, the model compiler cannot derive its type from an object flow. Instead it searches in the class diagram for a <<use>> dependency between the class owning the action script and the type to be created. For example, the following statement creates a local object ...

```
create local exchange using exchangeRate;
```

... whereas the <<use>> dependency **exchangeRate** is defined in a class diagram:

Figure: <<use>> Dependency Example



Note: This statement must be in an action script that is in the context of **LocalVariablesPort**, otherwise the <<use>> dependency **exchangeRate** cannot be found.

Example File (Builder project Basic Modeling/Data):

<your example path>\Basic Modeling\Data\uml\localVariables.xml

On this Page:

- Local Variables Declaration
- Local Creation of Complex Types
- Local Array Variables

However, sometimes we have to create types that are associated to a given object. In this case we can use the statement

```
create local nameOfLocalVariable using typeOf(expression);
```

This statement creates a complex local object having the same type as derived from the expression. For instance, if the object **obj** has an attribute **attr1** of type **MyType**, we can create a local variable **x** of type **MyType** by using the following statement.

```
create local x using typeOf(obj.attr1);
```

Additionally, it might be required to create arrays of complex types. Do this by stating

```
create local array nameOfLocalArray using dependencyNameToArrayElementType;
create local array nameOfLocalArray using typeOf(expression);
```

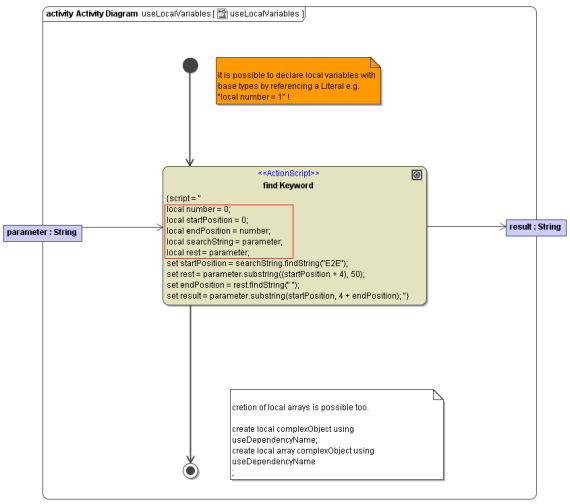
In this case we get arrays whose elements are of the type derived from ***dependencyName*** respectively ***expression***. The derived type is then the type of the array element.

Local Variables Declaration

Syntax	`local nameOfLocalVariable = <expression>;`	
Semantics	Declaring a local variable having the same type as expression.	
Substitutables	`nameOfLocalVariable`	Any given name.
	`expression`	Expressions are object accessors, operation calls, boolean expressions, literals, etc.

Examples	`local endPosition = myArray.count();`
	`local myObject = anotherObject.myObject;`
	`local mylocalinteger = 1;`
	`local myLocalString = "empty String";`
	`local anotherLocalString = myObject.myName;`	

Figure: Declaring Local Objects

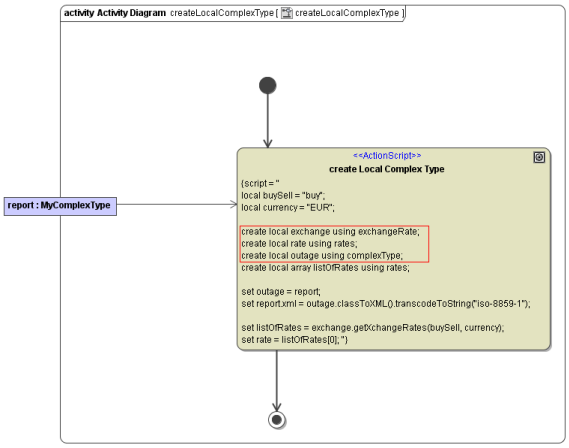


Local Creation of Complex Types

In order to create a local complex type, the following syntax is used:

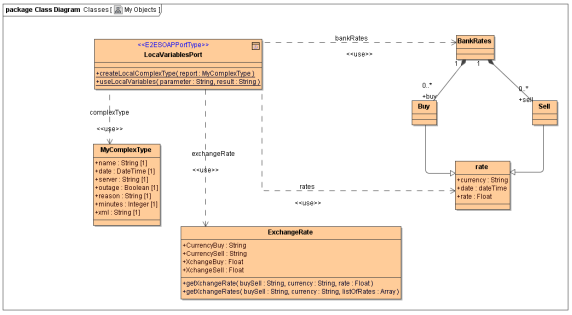
Syntax	`create local nameOfLocalVariable using dependency;` `create local nameOfLocalVariable using typeOf(expression);`	
Semantics	Creates a local variable of complex type usable within the action script.	
Substitutables	`nameOfLocalVariable`	Any given name.
	`dependencyName`	Name of a <<use>> dependency connecting the owning class of the action script to the type to be created.
	`expression`	Any valid action script expression that can be evaluated to a type.
Examples	`create local exchange using exchangeRate;` `create local tmp using typeOf(myObject.myAttribute);`	

Figure: Creating Complex Local Objects



To declare the used complex type, a <<use>> dependency from the class containing the action script (via an operation) to the used class must be drawn.

Figure: Defining <<use>> Dependencies



The use dependency must have a name. This name is used in the create statement after the **using** keyword.

Local Array Variables

In order to create a local array, the following syntax is used:

Syntax	<pre>create local array nameOfLocalArray using dependencyToArrayElementType;
	create local array nameOfLocalArray using typeOf (expression);</pre>	

Semantics	Creates a local array using the array element declared after the equal sign.	
Substitutables	`nameOfLocalArray`	Any given name.
	`dependencyToArrayElementType`	Name of a <<use>> dependency between the class containing the action script and the type of the array element.
	`expression`	Any valid action script expression that can be evaluated to a type. This is then the type of the array element.

Examples	<pre>create local array listOfRates using rates;
	create local array myArray using typeOf(myObject.myAttribute[1]);</pre>	

Figure: Defining Local Array Objects