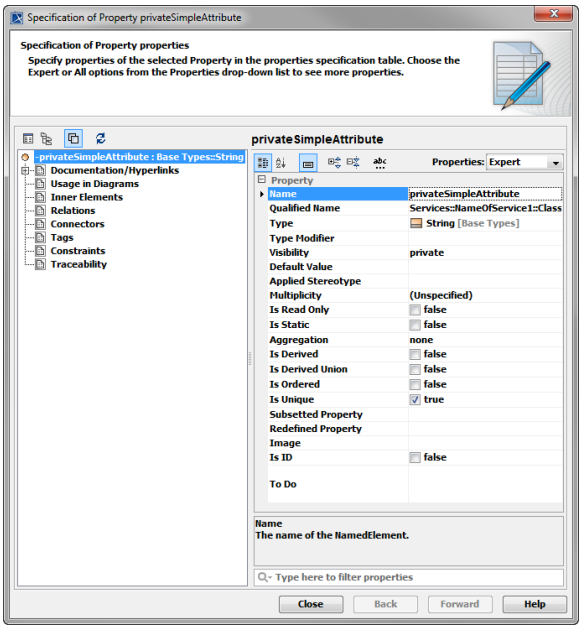
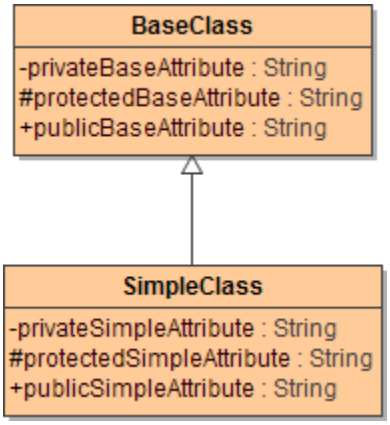
Attribute Specification

The figure below shows the attribute specification dialog to edit the properties of an attribute.

Figure: Attribute Specification Dialog





On this Page:

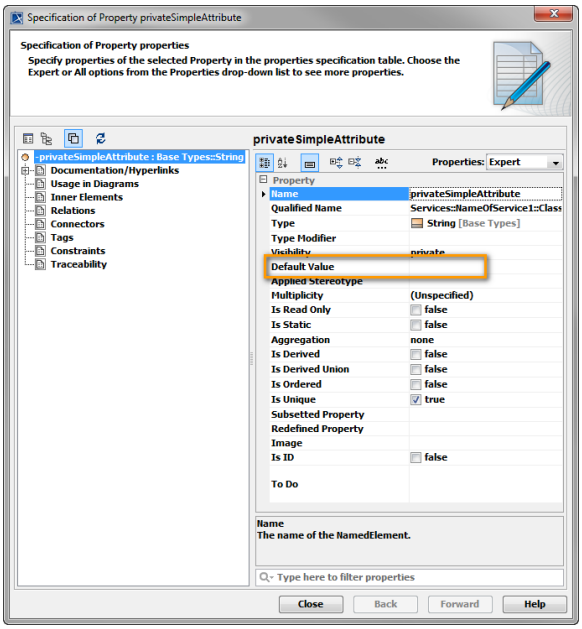
Related Pages:

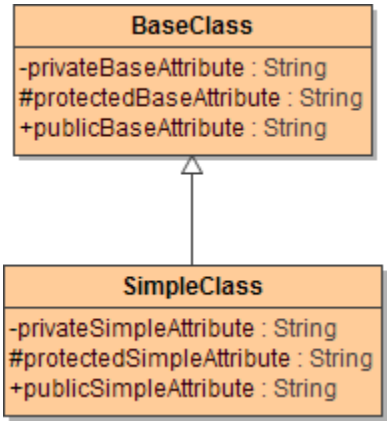
- setting() Macro
- xUML Service Settings
- Using Global Setting Variables
- Settings
- Class Design
- Associations

Initial Value

The figure below shows the usage of the property **initial value**. If an object of the class **SimpleClass** is instantiated (created) in the action script of an activity diagram, the attribute **privateSimpleAttribute** is assigned the value given in field **Default Value** automatically.

Figure: Initial Value Specification



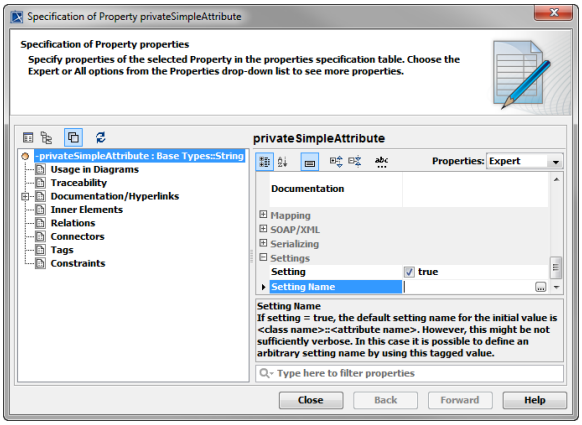


Default Value can be modified by right-clicking the field text, selecting **Value Specification** from the context menu and selecting one of the value types.
In addition to specifying a simple value, **Default Value** can also contain a global setting variable. See Using Global Setting Variables and setting() Macro for more information on that.

Frequently, it is necessary to store literals global to an xUML service. In Bridge context, it is possible to define name value pairs that are configurable on the Bridge. This can be achieved by applying the stereotype <<E2EAttribute>> to the corresponding attribute.
As soon as <<E2EAttribute>> is applied, additional specifications can be made in the **Settings** section of the specification dialog of the attribute.

Tagged Value	Description	Values	
Setting	Specify whether this attribute is a service setting. Service setting values can be configured via the Bridge, so that the default value of the attribute can be overridden.	true	Value is a setting and can be configured via the Bridge.
		false	Value is not a setting (default).
Setting Name	Specify an optional name of the setting to be displayed on the Bridge. If no name is specified, the Bridge uses `<<class name>>::<<attribute name>>`.	any valid string	
Is Password Setting	Specify whether the setting value contains a password that should not be displayed in the Bridge UI.	true	Value is a password and should be masked.
		false	Value is no password (default).

Figure: Specification Dialog with Settings Section



For more information on how to change the settings of an xUML service on the Bridge refer to xUML Service Settings.

Visibility

The visibility describes whether an attribute is accessible from a given context. The context is dependent on the owner of the activity diagram. For instance, if a class has an operation, which is implemented by an activity diagram, then the context is the class.
There are four attribute visibility types:

Visibility Type	Symbol	Description
public	+	The attribute is inherited and accessible by any outside object.
protected	-	The attribute is inherited and accessible from inside the current class (class operations) or classes derived from that class.
private	#	The attribute is not inherited and accessible only from inside the current class (class operations).
package	~	Not supported yet. The attribute's visibility will be handled like **public**.

For example, assume an activity diagram **Act1** is assigned to the operation **Op1** of class **Class1**. In this case, each attribute of class **Class1** can be used as input or output of adapters, functions, and EAL (E2E Action Language) expressions in this activity diagram. This works independently of the attributes' visibility, because the attributes of class **Class1** and the activity diagram **Act1** are in the same context. However, assume the class **Class1** is used in an activity diagram **Act2**, which is assigned to an operation **Op2**, which is a member of class **Class2**. All actions taking place in activity diagram **Act2** are in the context of class **Class2**. This has the following consequences:

- Within the activity diagram **Act2**, it is not possible to access **private** attributes of class **Class1**.
- If class **Class2** is inherited from class **Class1**, it is possible to access **protected** attributes of class **Class1**.

If an attribute is **public**, it is accessible from each context.
Using the visibility makes sense especially with stateful objects (persistent state objects) where the state variables should not be accessible from another class. If the class is used to model data structures, it is recommended to set the visibility of all attributes to **public**.
Classes can be modeled in a classical sense having properties (attributes) and behavior (operations). They are candidates for introducing getter and setter operations. For more details see Class Design.

Setting the Default Visibility of Attributes in a MagicDraw Project

Default visibility of newly created attributes in MagicDraw is private. When developing xUML services this can be annoying, because for each attribute user has to change the visibility to public.

You can change this behavior on a project basis. In MagicDraw, go to **Options > Project > Default Model Properties > Property**. Set the value of **Visibility** to **public**.

Hiding Attributes From Interfaces

Independent of the visibility specified, all attributes are exposed if a class is used in an interface.

Builder 7.5.0 You can hide selected attributes from interfaces. To do so, apply stereotype <<E2EPrivate>> on such attributes.

Attributes marked with<<E2EPrivate>> will

- not be exposed to WSDL, XSD, OpenAPI interfaces
- not be serialized to service responses for REST and SOAP
- not be serialized to requests of REST and SOAP adapters
- not be serialized when using classToXML() Operation and classToJSON() Operation.
- not be serialized and thus not saved between transitions and states, if applied to attributes of persistent state classes.

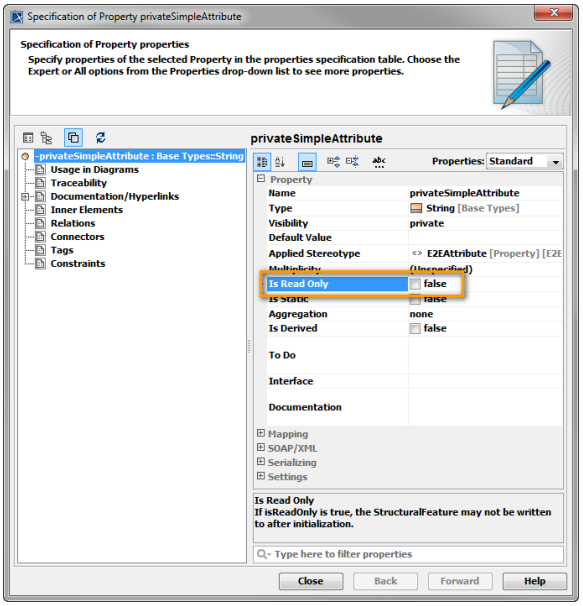
Changeability

An accessible attribute can be read or modified in the action script of an activity diagram depending on its attribute changeability settings.
There are three attribute changeability types:

- **changeable**: The attribute value can be modified.
- **frozen**: The attribute value may not be altered once the object is instantiated and its values initialized. No additional values may be added to a set of values.
- **add only**: This is meaningful only if the multiplicity is not fixed to a single value (e.g. arrays). Additional values may be added to the set of values, but once created a value may not be removed or altered.

This is can be implemented by the use of the property Read only.

Figure: Read Only Property



Scope

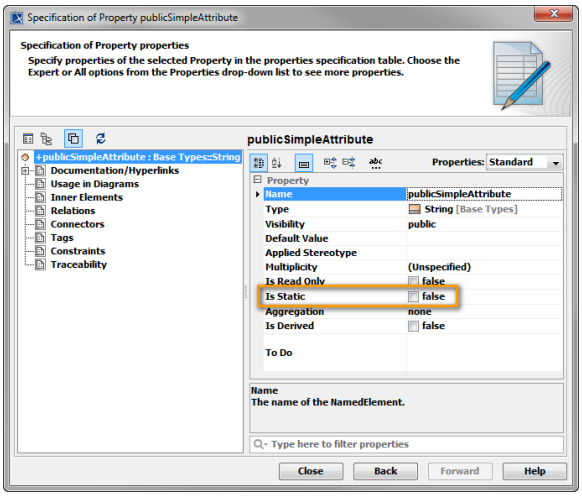
There are two possible scope types: **instance** and **classifier**.

If the scope of an attribute is **classifier**, the attribute will have the same value in all instances of the class. If the scope of an attribute is **instance**, the attribute values depend on the instances of the class.

> The Bridge does not support classifier scopes.

The attribute scope is defined by the flag **Is Static** in the attribute specification dialog. As classifier scopes are not supported, this flag remains always **false**.

Figure: Is Static Property



Multiplicity

Multiplicity describes the possible number of values of the attribute that may be held by an instance. The cardinality of the set of values is an implicit part of the attribute. In the common case, in which the multiplicity is **1..1**, the attribute is scalar (i.e., it holds exactly one value). If the multiplicity is for instance **0..** {*}, the attribute is an array, to which values can be appended.

The following multiplicities exist:

Multiplicity	Description
0	zero and only zero
1	one and only one
0..1	zero or one
0..*	from zero to any positive integer
1..*	from one to any positive integer
*	any positive integer

Above rules apply to associations as well.