Opaque objects

Macro Engine creates opaque objects to represent the elements in MagicDraw. Through these opaque objects, you can access the elements, retrieve, or assign values to them instead of using MagicDraw OpenAPI to do it.

All examples given in this section is written in Javascript.

On this page

Getting an Opaque Object

You can get an opaque object of an existing MagicDraw element by using either:

```
(i) AutomatonMacroAPI.getOpaqueObjectByPath(String path)

(ii) AutomatonMacroAPI.getOpaqueObject(Element element)
```

If the above methods cannot find the element, they will return null.

(i) getOpaqueObjectByPath(String path)

To use getOpaqueObjectByPath(String path), for example, type:

```
AutomatonMacroAPI.getOpaqueObjectByPath ("PackageA::Element2");
```

(ii) getOpaqueObject(Element element)

To use getOpaqueObject(Element element), for example, type:

```
var element = com.nomagic.magicdraw.uml.Finder.byQualifiedName().find(Application.getInstance().getProject(),
"PackageB::ClassB");
var a = AutomatonMacroAPI.getOpaqueObject(element);
```

```
You can also use two other methods to get an opaque object as follows:

     (iii) AutomatonMacroAPI.getModelData()
     (iv) AutomatonMacroAPI.getSelectedElementFromContainmentTree()
```

(iii) getModelData()

This method will obtain an opaque object of the model Data in the Containment tree.

(iv) getSelectedElementFromContainmentTree()
This method will obtain an opaque object of the selected element in the Containment tree.
Macro Engine uses methods (iii) getModelData() and (iv) getSelectedElementFromContainmentTree() to retrieve opaque objects in order to identify the defined scope in its recording mechanism.

Getting Element Property Values

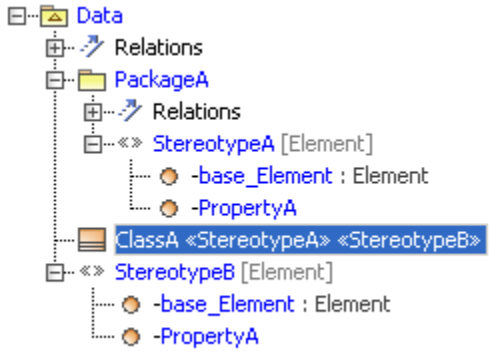
Once you have obtained an opaque object, you can get the property value of the element by using any of the following methods:

```
(i) <variableName>.get<PropertyName>()
(ii) <variableName>.<PropertyName>
(iii) <variableName>._automatonGetValue(String realPropertyName, String stereotypePath)
```

The <variableName> is the name of a macro variable that stores the opaque object. The <PropertyName> is the name of the property that appears in the Specification dialog.

The <variableName> is the name of a macro variable that stores the opaque object. The <PropertyName> is the name of the property that appears in the Specification dialog.

You need to capitalize the first letter of <PropertyName> and replace the whitespace with an underscore. If a duplicate property name occurs, you can refer to the right property name by using the following additional information: <SterotypeName>_<PropertyName><RunningNumber>.



If there are two stereotypes applied to the same element, see above figure, you can differentiate one from the other, for example, by specifying <PropertyName> as StereotypeA_PropertyA1 and StereotypeA_PropertyA2 in the macro.

You can also use the method _automatonGetValue to get a property value. If you want to get the value of a PropertyA from SterotypeA in PackageA., for example, you can use _automatonGetValue("PropertyA", "PackageA::StereotypeA").

A realPropertyName is the real property name that is used in MagicDraw openAPI. A stereotypePath is the path of a stereotype that contains the property. This property will not be needed if it is in the Element itself.

If you refer to a property that does not exist, Macro Engine may or may not throw an error, depending on which language library you use. For example, if you use Javascript to call the property that does not exist, Macro Engine will not throw an error. But if you use JRuby, it will throw an exception to report the error condition: org.jruby.exceptions.RaiseException.

If the value of a property is an element, Macro Engine will convert it to an opaque object and you can call it, for example, by typing classA.Owner.Name.

If a property has more than one value, Macro Engine will convert the values to a list of opaque objects. If you need to get a value from the list, you can call the method of the class java.util.List, for example, by typing:

```
(i) classA.Applied_Stereotype.get(<index>).Name
or
(ii) classA.Applied_Stereotype.add(anotherOpaque)
If a property is read-only, an exception will be thrown.
```

Getting Element Property Value Examples

The following are some examples of how to get an element property value using the methods given in the section above:
to use get<PropertyName>, for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath ("MyClass");
Application.getInstance().getGUILog().log(classA.getName());
```

to use <PropertyName>, for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("MyClass");
Application.getInstance().getGUILog().log(classA.Name);
```

to use _automatonGetValue, for example, type:

```
var reqA = AutomatonMacroAPI.getOpaqueObjectByPath("MyRequirements");
Application.getInstance().getGUILog().log(reqA._automatonGetValue("PropertyA", "PackageA::StereotypeA");
```

to use a SysML Element, for example, type:

```
var reqA = AutomatonMacroAPI.getOpaqueObjectByPath("MyRequirements");
Application.getInstance().getGUILog().log(reqA.getID());
```

Setting Element Property Values

You can assign values to a MagicDraw element by using any of the following methods:

```
(i) <variableName>.set<PropertyName>(Object value)
(ii) <variableName>.<PropertyName> = value; and then call persist()
(iii) <variableName>._automatonSetValue(String realPropertyName, String stereotypePath, Object value)
```

The value of an element can be a primitive data type, an opaque object, or an element. If you use a setter to set the value, for example, _automatonSetValue() or change the value on a list, it will be saved in the MagicDraw element automatically.

If you use <variableName>.<PropertyName> = value to set the value, you must call persist() to persist the change to the MagicDraw element. You need to first set the data, call persist(), and finally call a getter method in order to set the data and retrieve them. This process will force an opaque object to retrieve the current value from a MagicDraw model and overwrite the value that you have just specified in the opaque object.

If you use JRuby, do not capitalize the first letter of <PropertyName> in <variableName>.<PropertyName>.

Setting Element Property Value Examples

The following are some examples of how to set an element property value by using the methods given in section above.
to use set<PropertyName>(value), for example, type:

```
var classB = AutomatonMacroAPI.getOpaqueObjectByPath("Element2");
classB.setName("NewElementName")
```

to use <PropertyName> = value), for example, type:

```
var classB = AutomatonMacroAPI.getOpaqueObjectByPath("Element2");
classB.Name = "NewElementName";
classB.Is_Abstract = true;
classB.persist();
```

to use _automatonSetValue, for example, type:

```
var classB = AutomatonMacroAPI.getOpaqueObjectByPath("Element2");
classB._automatonSetValue("PropertyA", "PackageA::StereotypeA", "Demo String value");
```

to set an opaque object to another opaque object, for example, type:

```
var ele1 = AutomatonMacroAPI.getOpaqueObjectByPath("Element1");
var ele2 = AutomatonMacroAPI.getOpaqueObjectByPath("Element2");
ele1.setPackaged_Element(ele2);
```

The table below lists the supported element properties in Macro Engine.

Property	Type	Support Operation
Active Hyperlink	String	Read
All General Classifiers	List<AutomatonOpaqueObject>	Read
All Realizing Elements	List<AutomatonOpaqueObject>	Read

All Specific Classifiers	List<AutomatonOpaqueObject>	Read
All Specifying Elements	List<AutomatonOpaqueObject>	Read
Applied Stereotype	List<AutomatonOpaqueObject>	Read
Applied_Stereotype_Instance	N/A	Read
Attribute	List<AutomatonOpaqueObject>	Read
Base Classifier	List<AutomatonOpaqueObject>	Read
Class	AutomatonOpaqueObject	Read
Classifier Behavior	AutomatonOpaqueObject	Read
Client Dependency	List<AutomatonOpaqueObject>	Read
Collaboration Use	List<AutomatonOpaqueObject>	Read
Element	ID	N/A -
Extension	List<AutomatonOpaqueObject>	Read
Feature	List<AutomatonOpaqueObject>	Read
Generalization	List<AutomatonOpaqueObject>	Read
Image	N/A	-
Imported Member	List<AutomatonOpaqueObject>	Read
Inherited Member	List<AutomatonOpaqueObject>	Read
Interface Realization	List<AutomatonOpaqueObject>	Read
Is Leaf	Boolean	Read/Write
Is Final Specialization	Boolean	Read/Write
Is Active	Boolean	Read/Write
Is Abstract	Boolean	Read/Write
Member	List<AutomatonOpaqueObject>	Read
Name	String	Read/Write
Name Expression	AutomatonOpaqueObject	Read
Namespace	AutomatonOpaqueObject	Read
Nested Classifier	List<AutomatonOpaqueObject>	Read
Owned Attribute	List<AutomatonOpaqueObject>	Read
Owned Connector	List<AutomatonOpaqueObject>	Read
Owned Comment	List<AutomatonOpaqueObject>	Read
Owned Diagram	List<AutomatonOpaqueObject>	Read
Owned Element	List<AutomatonOpaqueObject>	Read
Owned Member	List<AutomatonOpaqueObject>	Read
Owned Operation	List<AutomatonOpaqueObject>	Read
Owned Port	List<AutomatonOpaqueObject>	Read
Owned Reception	List<AutomatonOpaqueObject>	Read
Owned Rule	List<AutomatonOpaqueObject>	Read
Owned Template Signature	AutomatonOpaqueObject	Read
Owned Trigger	List<AutomatonOpaqueObject>	Read
Owned Use Case	List<AutomatonOpaqueObject>	Read
Owning Package	List<AutomatonOpaqueObject>	Read

Owning Template Parameter	N/A	-
Owner	AutomatonOpaqueObject	Read/Write
Package	AutomatonOpaqueObject	Read
Package Import	List<AutomatonOpaqueObject>	Read
Part	List<AutomatonOpaqueObject>	Read
Participates In Activity	List<AutomatonOpaqueObject>	Read
Participates In Interaction	List<AutomatonOpaqueObject>	Read
Powertype Extent	List<AutomatonOpaqueObject>	Read
Qualified Name	List<AutomatonOpaqueObject>	Read
Realized Interface	N/A	-
Realizing Component	List<AutomatonOpaqueObject>	Read
Realizing Element	List<AutomatonOpaqueObject>	Read
Redefined Classifier	List<AutomatonOpaqueObject>	Read
Redefined Element	List<AutomatonOpaqueObject>	Read
Redefinition Context	List<AutomatonOpaqueObject>	Read
Representation	N/A	-
Role	List<AutomatonOpaqueObject>	Read
Specific Classifier	List<AutomatonOpaqueObject>	Read
Specifying Component	List<AutomatonOpaqueObject>	Read
Specifying Element	List<AutomatonOpaqueObject>	Read
Supplier Dependency	List<AutomatonOpaqueObject>	Read
Template Binding	List<AutomatonOpaqueObject>	Read
Template Parameter	N/A	-
To Do	String	Read/Write
Use Case	List<AutomatonOpaqueObject	Read/Write
Visibility	String	Read/Write

Getting the Child of an Element

You can get the child of an Element by typing the following:

- `<variableName>._getChild(String childElementName)`

If the above method cannot find the childElementName, it will throw an error.

To get the child of an element named ClassB from ClassA, for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("ClassA");
var classB = classA._getChild("ClassB");
```

Getting the Owner of an Element

You can get the owner of an element by typing the following:

- <variableName>._getOwner()

If the above method cannot find the owner, for example Model Data, it will throw an error.
To get the owner of an element named ClassA, for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("ClassA");
var classB = classA._getOwner();
```

Creating a New Element

You can create a new element by using its Meta-class name, such as a Class in Standard UML or a Requirement in SysML by using the following method:

- AutomatonMacroAPI.createElement(String metaClassName)

This method will return an opaque object of the created element. If the method cannot find the Meta-class, it will throw an exception.

To create a new Class & Requirement element (Javascript), for example, type:

```
AutomatonMacroAPI.createElement("Class");
AutomatonMacroAPI.createElement("Requirement");
```

Creating a Relationship Between Elements

You can create a relationship between elements by typing:

- AutomatonMacroAPI.createRelationship(AutomatonOpaqueObject opque1, AutomatonOpaqueObject opque2, String relationName)

To create a Copy & Abstraction relationship between Element1 and Element2 (Javascript), for example, type:

```
var a = AutomatonMacroAPI.getOpaqueObjectByPath("Element1");
var b = AutomatonMacroAPI.getOpaqueObjectByPath("Element2");
AutomatonMacroAPI.createRelationship(a, b, "Copy");
AutomatonMacroAPI.createRelationship(a, b, "Abstraction");
```

Removing an Element

You can remove an Element from MagicDraw by typing the following:

- AutomatonMacroAPI.removeElement(AutomatonOpaqueObject opaqueObj)

To remove an Element1 (Javascript), for example, type:

```
var a = AutomatonMacroAPI.getOpaqueObjectByPath("Element1");
AutomatonMacroAPI.removeElement(a);
```

Adding a Stereotype to an Element

You can apply a stereotype to an element by typing either:

(i) AutomatonMacroAPI.addStereotype(AutomatonOpaqueObject opaque, AutomatonOpaqueObject opaqueStereotype)
(ii)AutomatonMacroAPI.addStereotypeByStereotypeName(AutomatonOpaqueObject opaque, String stereotypeName)

To add a StererotypeA to ClassA (Javascript), for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("ClassA");
var stereotypeA = AutomatonMacroAPI.getOpaqueObjectByPath("StereotypeA");
AutomatonMacroAPI.addStereotype(classA, stereotypeA);
```

Removing a Stereotype from an Element

You can remove a Stereotype from an element by typing either:

(i) AutomatonMacroAPI.removeStereotype(AutomatonOpaqueObject opaque, AutomatonOpaqueObject opaqueStereotype)
(ii) AutomatonMacroAPI.removeStereotypeByStereotypeName(AutomatonOpaqueObject opaque, String stereotypeName)

To remove a StereotypeA from ClassA (Javascript), for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("ClassA");
var stereotypeA = AutomatonMacroAPI.getOpaqueObjectByPath("StereotypeA");
AutomatonMacroAPI.removeStereotype(classA, stereotypeA);
```

Printing Element Details

If you want to know the method that is used in the opaque object of an element, you can print the element details by typing the following:

- AutomatonMacroAPI.printElementDetail(AutomatonOpaqueObject opaque);

This method will print the field, constructor, and method that belong to the opaque object in the MagicDraw Message dialog.

To print the details of ClassA, for example, type:

```
var classA = AutomatonMacroAPI.getOpaqueObjectByPath("ClassA");
AutomatonMacroAPI.printElementDetail(classA);
```