What follows is a list of the OMC APIs with their syntax and examples.

- **getClassNames()**
  Returns the names of all class definitions in the global scope.
  Example

```
package test
    package test2
        model mymodel
        end mymodel;
    end test2;
    package test3
    end test3;
end test;
```

  Command: *getClassNames()*
  Reply: {test}

- **getClassNames(A1<cref>)**
  Returns the names of all class definitions in a class *A1*, according to the fact that in Modelica a package is a class.
  Example

```
package test
    package test2
        model mymodel
        end mymodel;
    end test2;
    package test3
    end test3;
end test;
```

  Command: *getClassNames(test)*
  Reply: {test2,test3}

- **getClassRestriction(A1<cref>)**
  Returns the kind of restricted class of *A1*, e.g. "model","connector", "function", "package", etc.
  Example

```
package test
    package test2
        model mymodel
        end mymodel;
    end test2;
    package test3
    end test3;
end test;
```

  Command: *getClassRestriction(test)*
  Reply: "package"

Command: *getClassRestriction(test.test2.mymodel)*
Reply: "model"

- **getErrorString()**
  Fetches the error string from OMC. This should be called after an "Error" is received

- **is*(A1<cref>)**
  Returns "true" if A1 is a Modelica class of type *, otherwise "false". The API of this family are: *isModel*, *isPackage*, *isPrimitive*, *isConnector*, *isRecord*, *isBlock*, *isType*, *isFunction*, *isClass*, *isParameter*, *isConstant*, *isProtected*.
  Example

```
package test
    package test2
        model mymodel
        end mymodel;
    end test2;
    package test3
    end test3;
end test;
```

  Command: *isPackage(test)*
  Reply: true

  Command: *isPackage(test.test2.mymodel)*
  Reply: false

- **getElementsInfo(A1<cref>)**
  Retrieves the Info attribute of all elements within the given class (A1). This contains information of the element type, filename, isReadOnly, line information, name etc., in the form of a vector containing element descriptors on record.
  Example
  In this example a model of Modelica library is used (supposing that the Standard library is already loaded in OMC).

  Command: *getElementsInfo(Modelica.Electrical.Analog.Example.ChuaCircuit)*
  Reply: { { rec(elementvisibility=public, elementfile = "MODELICALIBRARY/Modelica/Electric - al/Analog/Examples/ChuaCircuit.mo", elementreadonly="writable", elementStartLine=2, elementStart-Column=4, elementEndLine= 2, element End Column= 43, final=false, replaceable= false, inout= "none", element type= import, kind= qualified, path= Modelica.Electrical.Analog.Basic)}, {rec (element visibility= public, element file = "MODELICALIBRARY/Modelica/Electrical/Analog/Examples/ChuaCircuit.mo", elementreadonly= "writable", elementStartLine=3, elementStartColumn=4, elementEndLine =

3, elementEndColumn=56, final=false, replaceable= false, inout= "none", elementtype= import, kind= qualified, path= Modelica. Electrical. Analog. Examples. Utilities)} ,{rec (element visibility= public, elementfile= "MODELICALIBRARY/Modelica/Electrical/Analog/Examples/ChuaCircuit.mo", elementreadonly= "writable", elementStartLine=4, elementStartColumn= 4, element End Line=4, elementEndColumn=25, final= false, replaceable= false, inout="none", elementtype= import, kind= qualified, path= Modelica.Icons)}, {rec (elementvisibility= public, elementfile = "MODELICALIBRARY/Modelica/Electrical/Analog/Examples/ChuaCircuit.mo", elementreadonly= "writable", elementStartLine= 5, elementStartColumn= 4, elementEndLine=41, elementEndColumn= 5, final= false, replaceable= false, inout= "none", elementtype= extends, path= Icons.Example) },{rec (elementvisibility= public, elementtype= annotation) }, {rec(elementvisibility= public, elementfile = "MODELICALIBRARY/Modelica/Electrical/Analog/Examples/ChuaCircuit.mo", elementreadonly= "writable", elementStartLine=42, elementStartColumn=9, elementEndLine=42, elementEndColumn=190, final= false,replaceable= false, inout= "none", elementtype= component, typename= Basic.Inductor, names= {L,""}, flow=false, variability= "unspecified", direction= "unspecified") },...

This is not the complete answer but only a part of it because of its length.

- **getClassInformation(A1<cref>)**
  Returns a list of information about the class *A1*. The list is composed by: *{"restriction", "comment", "filename.mo", {boolean1, boolean2}, {"readonly| writable", integer1, integer2, integer3, integer4}}*.
  Restriction, *comment* and *filename* represent the restriction, the comment and the file name of the class *A1*. The boolean values tell if *A1* is partial, if it is a final or it is encapsulated. The third element is *"readonly"* if the class can be read or *"writable"* if it can be modified; the four integer values represent the start line, start column, end line and end column of *A1* in *filename.mo*.

- **getIconAnnotation(A1<className>)**
  Returns the Icon Annotation of the class named *A1*. The result is the flattened code of the actual annotation of the class. Since the Diagram annotations can be found in base classes, a partial code instantiation is performed that flattens the inheritance hierarchy in order to find all annotations. Because of the partial flattening, the format returned is not according the Modelica standard for Diagram annotations.
  Example

  Command: *getIconAnnotation(Modelica.Electrical.Analog.Basic.Resistor)*
  Reply: {-100.0, -100.0, 100.0, 100.0, {Rectangle (true, {0,0,255}, {255,255,255}, LinePattern.Solid, FillPattern.Solid, 0.25, BorderPattern.None, {{-70.0, 30.0}, {70.0, -30.0}}, 0.0), Line (true, {{-90.0, 0.0}, {-70.0, 0.0}}, {0,0,255}, LinePattern.Solid,0.25, {Arrow.None,Arrow.None}, 3.0,false), Line(true, {{70.0,0.0}, {90.0,0.0}}, {0,0,255}, LinePattern.Solid,0.25, {Arrow.None, Arrow.None}, 3.0, false), Text(true, {0,0,0}, {0,0,0}, LinePattern.Solid, FillPattern.None, 0.25, {{-144.0, -60.0}, {144.0,-100.0}},

"R=%R", 0.0, ""), Text (true, {0,0,0}, {0,0,255}, LinePattern.Solid, FillPattern.None, 0.25, {{-144.0, 40.0}, {144.0,100.0}} ,"%name", 0.0, "")}}

- **getDiagramAnnotation(A1<className>)**
  Returns the Diagram Annotation of the class named *A1*. The result is the flattened code of the actual annotation of the class. Since the Diagram annotations can be found in base classes a partial code instantiation is performed that flattens the inheritance hierarchy in order to find all annotations. Because of the partial flattening, the format returned is not according the Modelica standard for Diagram annotations.
  Example

  Command: *getDiagramAnnotation(Modelica.Electrical.Analog.Basic.Resistor)*
  Reply: {-100.0, -100.0, 100.0, 100.0, {Rectangle (true, {0, 0, 255}, {0,0,0}, LinePattern.Solid, FillPattern.None, 0.25, BorderPattern.None, {{-70.0,30.0}, {70.0,-30.0}}, 0.0), Line (true, {{-96.0,0.0}, {-70.0, 0.0}}, {0,0,255}, LinePattern.Solid, 0.25, {Arrow.None,Arrow.None}, 3.0, false), Line(true, {{70.0,0.0}, {96.0,0.0}}, {0,0,255}, LinePattern.Solid,0.25, {Arrow.None,Arrow.None}, 3.0, false)}}

- **getDocumentationAnnotation(A1<cref>)**
  Returs the Documentation Annotation of the class named *A1*.
  Example

  Command: *getDocumentationAnnotation(Modelica.Electrical.Analog.Basic.Resistor)*
  Reply: {“<HTML>
  <P>
  The linear resistor connects the branch voltage v with the
  branch current i by i*R = v.
  The Resistance R is allowed to be positive, zero, or negative.
  </P>
  </HTML>
  ”}

- **loadFile(A1<string>)**
  Loads all models in the file *A1*.
  Example

  Command: *loadFile("/home/user/Desktop/model.mo")*
  Reply: true

- **loadModel(A1<cref>)**
  Loads the model *A1* by looking up the correct file to load in $OPENMOD-

ELICALIBRARY. Loads all models in that file into the symbol table.
Example

Command: *loadModel(Modelica)*
Reply: true

- **createModel(A1<cref>)**
  Creates a new empty model named *A1* in global scope. This method doesn't write any file but creates the model in OMC local memory, thus invoking *save(A1)* will return false. In order to save the new model the user has to link the new model to a file by *setSourceFile(A1<string>,A2<string>)*.
  Example

  Command: *getClassNames()*
  Reply: {}

  Command: *createModel(myModel)*
  Reply: true

  Command: *getClassNames()*
  Reply: {myModel}

  Command: *save(myModel)*
  Reply: false

  Command: *setSourceFile(myModel, "/home/user/filename.mo")*
  Reply: Ok

  Command: *save(myModel)*
  Reply: true

  Here is the content of the *filename.mo*

  ```
  model myModel
  end myModel;
  ```

- **newModel(A1<cref>, A2<cref>)**
  Creates a new empty model named *A1* in class *A2*. This method doesn't write any file but creates the model in OMC local memory, thus invoking *save(A1)* will return false. In order to save the new model the user has to link the new model to a file by *setSourceFile(A1<string>,A2<string>)*.
  Example

  Command: *package test end test;*
  Reply: Ok

  Command: *newModel(newModel, test)*
  Reply: true

Command: *getClassNames(test)*
Reply: {newModel}

Command: *save(test.myModel)*
Reply: false

Command: *setSourceFile(test.myModel, "/home/user/filename.mo")*
Reply: Ok

Command: *setSourceFile(test, "/home/user/filename.mo")*
Reply: Ok

Command: *save(test)*
Reply: true

Here is the content of the *filename.mo*

```
package test
    model myModel
    end myModel;
end test;
```

- **save(A1<cref>)**
  Saves the model *A1* into the file it was previously linked to.
  Example

  Command: *save(test.myModel)*
  Reply: true

- **deleteClass(A1<cref>)**
  Deletes the class from the symbol table.
  Example

  Command: *package test package test2 end test2; end test;*
  Reply: Ok

  Command: *deleteClass(test)*
  Reply: true

  Command: *getClassNames()*
  Reply: {}

- **renameClass(A1<cref>, A2<cref>)**
  Renames an already existing class with name *A1* to name *A2*. The rename is performed recursively in all already loaded models which reference the class *A1*. While *A1* can be in a dotted annotation form in order to refer to nested classes, *A2* can't since it represent the new identifier and not the path of the

class.

Example

Command: *package test package test2 end test2; package test3 end test3; end test;*
Reply: Ok

Command: *getClassNames()*
Reply: {test}

Command: *getClassNames(test)*
Reply: {test2,test3}

Command: *renameClass(test, newTest)*
Reply: {newTest}

Command: *getClassNames()*
Reply: {newTest}

Command: *renameClass(newTest.test2, newTest.test3.test6)*
Reply: error

Command: *renameClass(newTest.test2, test6)*
Reply: {newTest.test6}

Command: *getClassNames(newTest)*
Reply: {test6,test3}

- **setClassComment(A1<cref>,A2<string>)**

  Sets the class *A1* string comment *A2*. Notice that *A2* must be include into quotes.

  Example

```
package test
    model myModel
    end myModel;
end test;
```

  Command: *setClassComment(test.myModel, comment) /*malformed command*/*
  Reply:

  Command: *setClassComment(test.myModel, "this is a comment") /*notice the quotes*/*
  Reply: Ok

```
package test
    model myModel "this is a comment"
    end myModel;
end test;
```

- **addClassAnnotation(A1<cref>, annotate=<expr>)**
  Adds annotation given by <*expr*> (the second parameter must be in the form annotate=classmod(...)) to the model definition referenced by *A1*. It should be used to add Icon Diagram and Documentation annotations.
  Example

```
model mymodel
end mymodel;
```

  Command: *addClassAnnotation(mymodel, annotate= Icon(coordinateSystem= CoordinateSystem (extent= {{-100, -100}, {100, 100}}), graphics={}))*
  Reply: true

  Command: *save(mymodel)*
  Reply: true

```
model mymodel annotation(Icon(coordinateSystem
    (extent={{-100,-100},{100,100}}), graphics={}));
end mymodel;
```

  Command: *addClassAnnotation(mymodel, annotate= Icon(coordinateSystem= CoordinateSystem (extent= {{-100, -100}, {100, 100}}), graphics= {Line (color= {127,127,127}, arrow= {Arrow.none, Arrow.start}, points= {{-50, -50}, {50,50}, {100, 0}, {0, 100}}) }))*
  Reply: true

```
model mymodel annotation(Icon (coordinateSystem (extent= {{-100, -100},
    {100, 100}}),
    graphics= {Line(color= {127, 127, 127}, arrow= {Arrow.none, Arrow.st-
    art}, points= {{-50, -50},{50, 50},{100, 0},{0, 100}})}));
end mymodel;
```

  Command: *addClassAnnotation(mymodel, annotate= Diagram (coordinateSystem= CoordinateSystem (extent= {{-200, -150}, {10, 105}}), graphics= {Rectangle (lineColor= {127, 127, 127}, extent={{-20, -20}, {10, 15}}, pattern= LinePattern.DashDotDot), Text(extent= {{-5, 5},{50, 55}}, textString= "hello") }))*
  Reply: true

```
model mymodel annotation (Icon (coordinateSystem (extent= {{-100, -100},{100,
    100}}), graphics= {Line(color={127,127,127}, arrow={Arrow.none, Arrow.sta-
    rt}, points={{-50, -50},{50, 50},{100, 0}, {0, 100}})}), Diagram
    (coordinateSystem (extent= {{-200, -150},{10, 105}}), graphics={Rectangle
    (lineColor= {127, 127, 127}, extent={{-20,-20},{10,15}}, pattern=
    LinePattern.DashDotDot), Text(extent= {{-5, 5}, {50, 55}}, textString=
    "hello")}));
end mymodel;
```

```
model mymodel
end mymodel;
```

Command: *addClassAnnotation(mymodel, annotate=Diagram())*
Reply: true

```
model mymodel annotation(Documentation(info="<HTML>Hello</HTML>"));
end mymodel;
```

- **getPackages()**
  Returns the names of all package definitions in the global scope.
  Example

```
package test
    package test_1
    end test_1;
    package test_2
    end test_2;
    model model_1
    end model_1;
end test;
package test2
end test2;
```

Command: *getPackages()*
Reply: {test2,test}

- **getPackages(A1<cref>)**
  Returns the names of all Packages in a class/package named by *A1* as a list.
  Example

```
package test
    package test_1
    end test_1;
    package test_2
    end test_2;
    model model_1
    end model_1;
end test;
package test2
end test2;
```

Command: *getPackages(test)*
Reply: {test_1,test_2}

- **getClassAttributes(A1<cref>)**
  Returns all the possible information of class *A1* in the following form:
  rec(attr1 = value1, attr2 = value2 ...).
  Example

```
package test
    model mymodel
        Real x;
        Real y;
    equation
        x=y;
    end mymodel;
end test;
```

Command: *getClassAttributes(test.mymodel)*
Reply: { rec(name=“mymodel”, partial=false, final=false, encapsulated=false, restriction=MODEL, comment="", file="/home/ilcava/Desktop/modello.mo", readonly="writable", startLine= 2, startColumn= 25, endLine= 7, endColumn= 36) }

- **existClass(A1<cref>)**

  Returns *"true"* if class *A1* exists in symbolTable, *"false"* otherwise.

  Example

```
package test
    model mymodel
    end mymodel;
end test;
```

  Command: *existClass(test.mymodel)*
  Reply: true

- **existPackage(A1<cref>)**

  Returns *"true"* if package *A1* exists in symbolTable, *"false"* otherwise.

  Example

```
package test
    model mymodel
    end mymodel;
end test;
```

  Command: *existPackage(test)*
  Reply: true

- **existModel(A1<cref>)**

  Returns *"true"* if model *A1* exists in symbolTable, *"false"* otherwise.

  Example

```
package test
    model mymodel
    end mymodel;
end test;
```

  Command: *existModel(test.mymodel)*
  Reply: true

- **getComponents(A1<cref>)**
  Returns a list of component declarations within class *A1*: "{{Atype, varidA, "commentA"}, {Btype, varidB, "commentB"}, {...}}" and so on.
  Example

  Command: *getComponents(Modelica.Electrical.Analog.Examples.ChuaCircuit)*
  Reply: {{Modelica.Electrical.Analog.Basic.Inductor, L, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Basic.Resistor, Ro, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Basic.Conductor, G, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Basic.Capacitor, C1, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Basic.Capacitor, C2, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Examples.Utilities.NonlinearResistor, Nr, "", "public", false, false, false, "unspecified", "none", "unspecified"},
  {Modelica.Electrical.Analog.Basic.Ground, Gnd, "", "public", false, false, false, "unspecified", "none", "unspecified"}}}

- **setComponentProperties(A1 <cref>, A2 <cref>, A3 <Boolean>, A4 <Boolean>, A5 <Boolean>, A6 <Boolean>, A7 <String>, A8 <{Boolean, Boolean}>, A9 <String>)**
  Sets properties of component *A2* in a class *A1*. The properties are:

  - *A3* final (true/false)
  - *A4* flow (true/false)
  - *A5* protected(true) or public(false)
  - *A6* replaceable (true/false)
  - *A7* variability: "constant" or "discrete" or "parameter" or""

  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
    end mymodel;
end test;
```

  Command: *setComponentProperties(test.mymodel, r1, {true, true, true, false}, {"discrete"}, {true, false}, {"input"})*
  Reply: OK

```
package test
    model mymodel
        protected
final inner flow discrete input Modelica.Electrical.Analog.Basic.Resistor
    r1;
    end mymodel;
end test;
```

- **setComponentComment(A1<cref>, A2<cref>, A3<string>)**
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
    end mymodel;
end test;
```

  Command: *setComponentComment(test.mymodel, r1, "comment")*
  Reply: Ok

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1 "comment";
    end mymodel;
end test;
```

- **getComponentAnnotations(A1<cref>)**
  Returns a list of all annotations of all components in class *A1*, in the same order as the components, one annotation per component.
  Example

  Command: *getComponentAnnotations(Modelica.Electrical.Analog.Examples.ChuaCircuit)*
  Reply: { {true, -75.0, 50.0, 0.25, 1.0, false, false, -90.0, -75.0, 50.0, 0.25, 1.0, false, false, -90.0},
  {true, -75.0, -5.0, 0.25, 1.0, false, false, -90.0, -75.0, -5.0, 0.25, 1.0, false ,false, -90.0},
  {true,0.0,75.0,0.25,1.0,false,false,0.0,0.0,75.0,0.25,1.0,false,false, 0.0},
  {true,25.0,15.0,0.25,1.0,false,false,-90.0,25.0,15.0,0.25,1.0,false,false, -90.0},
  {true,-25.0,15.0,0.25,1.0,false,false,-90.0,-25.0,15.0,0.25,1.0,false,false, -90.0},
  {true,75.0,15.0,0.25,1.0,false,false,-90.0,75.0,15.0,0.25,1.0,false,false, -90.0},
  {true,0.0,-75.0,0.25,1.0,false,false,0.0,0.0,-75.0,0.25,1.0,false,false, 0.0}}

- **addComponent(A1<ident>,A2<cref>,A3<cref>,annotate=<expr>)**
  Adds a component with name *A1*, type *A2*, and class *A3* as arguments. Optional annotations are given with the named argument annotate.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
    end mymodel;
end test;
```

Command: *addComponent(c1, Modelica.Electrical.Analog.Basic.Capacitor, test.mymodel, annotate=Code(()))*
Reply: true

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    end mymodel;
end test;
```

Command: *addComponent(c2, Modelica.Electrical.Analog.Basic.Capacitor, test.mymodel, annotate=Placement(transformation=transformation(x=10, flipVertical=true), iconTransformation=transformation(y=5, scale=0.1, aspectRatio=1.2, rotation=-90, flipHorizontal=true)))*
Reply: true

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
        Modelica.Electrical.Analog.Basic.Capacitor c2 annotation (Pla-
    cement (transformation (x=10, flipVertical=true), iconTransformation
    (y=5, scale=0.1, aspectRatio=1.2, rotation=-90, flipHorizontal=true)));
    end mymodel;
end test;
```

- **deleteComponent(A1<ident>,A2<cref>)**

  Deletes a component *A1* within a class *A2*.

  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
        Modelica.Electrical.Analog.Basic.Capacitor c2
annotation (Placement (transformation (x=10, flipVertical= true), i-
    conTransformation (y=5, scale=0.1, aspectRatio=1.2, rotation=-90,
    flipHorizontal= true)));
    end mymodel;
end test;
```

  Command: *deleteComponent(c2, test.mymodel)*
  Reply: true

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    end mymodel;
end test;
```

- **updateComponent(A1<ident>,A2<cref>,A3<cref>,annotate=<expr>)**
  Updates an already existing component with name *A1*, type *A2*, and class *A3* as arguments. Optional annotations are given with the named argument annotate.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    end mymodel;
end test;
```

  Command: *updateComponent(c1, Modelica.Electrical.Analog.Basic.Capacitor, test.mymodel, annotate=Placement (transformation=transformation (x=25, scale=0.1, aspectRatio=1.2, rotation=-90), iconTransformation=transformation (y=5, flipVertical=true, flipHorizontal=true)))*
  Reply: true

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1 annotation (Placement
    (transformation (x=25, scale=0.1, aspectRatio=1.2, rotation=-90),
    iconTransformation (y=5, flipVertical=true, flipHorizontal=true)));
    end mymodel;
end test;
```

- **renameComponent(A1<cref>,A2<ident>,A3<ident>)**
  Renames an already existing component with name *A2* defined in a class with name *A1*, to the new name *A3*. The rename is performed recursively in all already loaded models which reference the component declared in class *A2*.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
    end mymodel;
end test;
```

  Command: *renameComponent(test.mymodel, r1, newName)*
  Reply: {test.mymodel}

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor newName;
    end mymodel;
end test;
```

- **getNthComponentAnnotation(A1<cref>,A2<int>)**
  Returns the flattened annotation record of the nth component *A2* (the first is has no 1) within class/component *A1*. It consists of a comma separated string of 15 values.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1 annotation (Pla-
    cement (transformation (x=100, y=100, scale=0.1, rotation=-90), i-
    conTransformation (x=10, y=50, flipVertical=true, scale=0.1, ro-
    tation= 90)));
        Modelica.Electrical.Analog.Basic.Capacitor c1 annotation (Pla-
    cement (transformation (x=25, scale=0.1, aspectRatio=1.2, rotation=
    -90), iconTransformation(y=5, flipVertical=true, flipHorizontal=true)))
    end mymodel;
end test;
```

  Command: *getNthComponentAnnotation(test.mymodel, 1)*
  Reply: {{true, 100.0, 100.0, 0.1, 1.0, false, false, -90.0, 10.0, 50.0, 0.1, 1.0, false, true,9 0.0}}

  Command: *getNthComponentAnnotation(test.mymodel, 2)*
  Reply: {{true, 25.0, 0.0, 0.1, 1.2, false, false, -90.0, 0.0, 5.0, 1.0, 1.0, true, true, 0.0}}

- **getNthComponentModification(A1<cref>,A2<int>)**
  Returns the modification of the nth component (of index *A2*) of class/component *A1*. The fist component has index 1.

- **getComponentModifierValue(A1<cref>, A2<cref)**
  Returns the value of a component (e.g. variable, parameter, constant, etc.) *A2* in a class A1.

- **setComponentModifierValue(A1<cref>,A2<cref>,A3<exp>)**
  Sets the modifier value of a component (e.g. variable, parameter, constant, etc.) *A2* in a class *A1* to an expression (unevaluated) in *A3*.

- **getComponentModifierNames(A1<cref>, A2<cref>)**
  Retrieves the names of all components in the class.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    end mymodel;
end test;
```

Command: *getNthComponentModification(test.mymodel, 1)*
Reply: {Code()}

Command: *setComponentModifierValue(test.mymodel, r1, Code(=2))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1=2;
...
```

Command: *getNthComponentModification(test.mymodel, 1)*
Reply: {Code(=2)}

Command: *setComponentModifierValue(test.mymodel, r1.start, Code(=2))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1(start=2)=2;
...
```

Command: *setComponentModifierValue(test.mymodel, r1, Code(=Resistor(R=2)))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1(start=2)=Resistor(R=2);
...
```

Command: *getNthComponentModification(test.mymodel, 1)*
Reply: {Code((start=2)=Resistor(R=2))}

Command: *setComponentModifierValue(test.mymodel, r1.min, Code(=10))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1(start=2, min=10)=Resistor(R=2);
...
```

Command: *getComponentModifierNames(test.mymodel, r1)*
Reply: {start, min}

Command: *getNthComponentModification(test.mymodel, 1)*
Reply: {Code((start=2, min=10)=Resistor(R=2))}

Command: *getComponentModifierValue(test.mymodel, r1)*
Reply: Resistor(R=2)

Command: *getComponentModifierValue(test.mymodel, r1.start)*
Reply: =2

Command: *getComponentModifierValue(test.mymodel, r1.min)*
Reply: =10

Command: *setComponentModifierValue(test.mymodel, r1.min, Code(()))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1(start=2)=Resistor(R=2);
...
```

Command: *setComponentModifierValue(test.mymodel, r1, Code(()))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1(start=2);
...
```

Command: *setComponentModifierValue(test.mymodel, r1.start, Code(()))*
Reply: Ok

```
...
Modelica.Electrical.Analog.Basic.Resistor r1;
...
```

- **getInheritanceCount(A1<cref>)**

  Returns the number (as a string) of inherited classes of a class *A1*.

  Example

  Command: *getInheritanceCount(Modelica.Electrical.Analog.Basic.Resistor)*
  Reply: 1

- **getNthInheritedClass(A1<cref>,A2<int>)**

  Returns the name of the nth inherited class of a class *A1*. The first class has number 1.

  Example

  Command: *getNthInheritedClass(Modelica.Electrical.Analog.Basic.Resistor, 1)*
  Reply: Modelica.Electrical.Analog.Interfaces.OnePort

- **getConnectionCount(A1<cref>)**
  Returns the number (as a string) of connections in the model *A1*.
  Example

  Command: *getConnectionCount(Modelica.Electrical.Analog.Examples.ChuaCircuit)*
  Reply: 9

- **setConnectionComment(A1<cref>, A2<cref>, A3<cref>, A4<string>**
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    equation
        connect(r1.p, c1.n);
    end mymodel;
end test;
```

  Command: *setConnectionComment(test.mymodel, r1.p, c1.n, "comment")*
  Reply: Ok

```
...
connect(r1.p,c1.n) "comment";
...
```

- **getNthConnection(A1<cref>,A2<int>)**
  Returns the nth connection declared in model *A1*, as a comma separated pair of connectors. The first has number 1.
  Example

  Command: *getNthConnection(Modelica.Electrical.Analog.Examples.ChuaCircuit, 2)*
  Reply: {G.n,Nr.p, ""}

- **getNthConnectionAnnotation(A1<cref>,A2<int>)**
  Returns the annotation of the nth connection of model *A1* as comma separated list of values of a flattened record.
  Example

  Command: *getNthConnectionAnnotation(Modelica.Electrical.Analog.Examples.ChuaCircuit, 2)*
  Reply: {Line (true, {{25.0, 75.0},{75.0, 75.0}, {75.0, 40.0}}, {0, 0, 255}, LinePattern.Solid, 0.25,{Arrow.None, Arrow.None}, 3.0, false)}

- **addConnection(A1<cref>,A2<cref>A3<cref>, annotate=<expr>)**
  Adds connection *connect(A1,A2)* to model *A3*, with annotation given by the named argument annotate.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    equation
        connect(r1.p,c1.n);
    end mymodel;
end test;
```

  Command: *addConnection(r1.p, c1.n, test.mymodel, annotate= Line (color= {127, 127, 127}, points={{10, 50},{50, 50},{50, 100}}))*
  Reply: Ok

```
...
connect(r1.p,c1.n) annotation(Line (color= {127, 127, 127}, points=
    {{10, 50}, {50, 50},{50,100}}));
...
```

  Command: *addConnection(r1.n, c1.n, test.mymodel, annotate="")*
  Reply: Ok

```
...
connect(r1.n,c1.n);
connect(r1.p,c1.n) annotation(Line (color= {127, 127, 127}, points=
    {{10, 50}, {50, 50},{50,100}}));
...
```

- **deleteConnection(A1<cref>,A2<cref>,A3<cref>)**
  Deletes the connection *connect(A1,A2)* in class *A3*.
  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    equation
        connect(r1.n,c1.n);
        connect(r1.p,c1.n) annotation(Line(color={127,127,127}, po-
    ints= {{10, 50}, {50, 50},{50, 100}}));
    end mymodel;
end test;
```

  Command: *deleteConnection(r1.n, c1.n, test.mymodel)*
  Reply: Ok

```
package test
```

```
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    equation
        connect(r1.p,c1.n) annotation(Line(color={127,127,127}, po-
    ints= {{10, 50}, {50, 50},{50, 100}}));
    end mymodel;
end test;
```

- **updateConnection(A1<cref>,A2<cref>,A3<cref>,annotate=<expr>)**

  Updates an already existing connection.

  Example

```
package test
    model mymodel
        Modelica.Electrical.Analog.Basic.Resistor r1;
        Modelica.Electrical.Analog.Basic.Capacitor c1;
    equation
        connect(r1.p,c1.n) annotation(Line(color={127,127,127}, po-
    ints= {{10, 50}, {50, 50},{50, 100}}));
    end mymodel;
end test;
```

  Command: *updateConnection(r1.p, c1.n, test.mymodel, annotate=Line (color={30,25,225}, points={{20, 10},{20, 150}}, pattern=LinePattern.Dot))*
  Reply: Ok

```
...
connect(r1.p,c1.n) annotation(Line (color= {30, 25, 225}, points={{20,
    10},{20, 150}},
pattern=LinePattern.Dot));
...
```

- **cd()**

  Returns current working directory.

  Example

  Command: *cd()*
  Reply: "/home/user/OpenModelica/share/openmodelica-1.4-dev"

- **cd(A1<string>)**

  Changes directory. Example

  Command: *cd("/home/user/work")*
  Reply: "/home/user/work"

- **checkModel(A1<cref>)**

  Instantiates model, optimizes equations, and reports errors. Example

Command: *checkModel(Modelica.Electrical.Analog.Examples.ChuaCircuit)*
Reply: "Check of Modelica.Electrical.Analog.Examples.ChuaCircuit successful.

model Modelica.Electrical.Analog.Examples.ChuaCircuit has 38 equation(s) and 38 variable(s). 23 of these are trivial equation(s)."

- **clear()**

  Clears everything: symbol table and variables.

  Example

  Command: *clear()*
  Reply: true

- **list(A1<cref>)**

  Prints class definition of class *A1*.

  Example

  Command: *list(Modelica.Electrical.Analog.Examples.ChuaCircuit)*
  Reply: "encapsulated model ChuaCircuit "Chua's circuit, ns, V, A"
  import Modelica.Electrical.Analog.Basic;
  import Modelica.Electrical.Analog.Examples.Utilities;
  import Modelica.Icons;
  extends Icons.Example;
  annotation (Diagram(coordinateSystem (extent={{-100.0, -100.0}, {100.0, 100.0}})),
  Icon (coordinateSystem (extent= {{-100.0, -100.0}, {100.0, 100.0}})), Documentation (info="<html>. . . </HTML>"));
  Basic.Inductor L(L=18) annotation (Placement (transformation (x= -75.0, y= 50.0, scale= 0.25, aspectRatio=1.0, rotation=-90), iconTransformation(x= -75.0, y= 50.0, scale= 0.25, aspectRatio=1.0, rotation= -90))); . . .

  equation
  connect(L.p,G.p) annotation(Line(points={{-75.0,75.0},{-25.0,75.0}}, color= {0, 0, 255})); . . .

- **getParameterNames(A1<cref>)**

  Gets the names all parameters of class *A1*.

  Example

```
package test
    model mymodel
        parameter Real p1;
        parameter Real p2;
    end mymodel;
end test;
```

Command: *getParameterNames(test.mymodel)*
Reply: {p1, p2}

- **setParameterValue (A1<cref>, A2<cref>, A3<cref>)**
  Sets the value of parameter *A2* in class *A1* to value *A3*.
  Example

```
package test
    model mymodel
        parameter Real p1;
        parameter Real p2;
    end mymodel;
end test;
```

  Command: *setParameterValue(test.mymodel, p1, 2)*
  Reply: Ok

```
...
parameter Real p1=2;
...
```

- **getParameterValue(A1<cref>, A2<cref>)**
  Gets the value of parameter *A2* in class *A1*.
  Example

```
package test
    model mymodel
        parameter Real p1=2;
        parameter Real p2;
    end mymodel;
end test;
```

  Command: *getParameterValue(test.mymodel, p1)*
  Reply: 2