Instructions to install and run the OSLC MagicDraw SysML Adapter (for v17.0.3, v17.0.4, v17.0.5)

By Axel Reichwein (axel.reichwein@koneksys.com) August 22, 2014

1. Installing Eclipse Lyo

Follow the instructions in the document named "Instructions to install Eclipse Lyo". The document also contains instructions on how to use a proxy server with Maven and Eclipse.

2. Downloading org.eclipse.lyo.adapter.magicdraw repository

1. Open the Git Repositories View (Window -> Show View -> type "Git Repositories" in the search field). The view may be displayed at the right or in the lower middle section of your Eclipse IDE
2. Click on the Clone Repository icon
3. In the URI field, paste the following URL: git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.adapter.magicdraw.git
4. The Host and Repository fields will autofill. Leave the Username and Password fields empty.
5. Click Next, Next again, and then Finish.

3. Importing projects into the Eclipse workspace

1. In the Git repositories view, right-click org.eclipse.lyo.adapter.magicdraw and select "Import Projects". Click Next until Finish
2. 3 projects additional will be in the Eclipse workspace

4. Choosing the MagicDraw-specific pom file

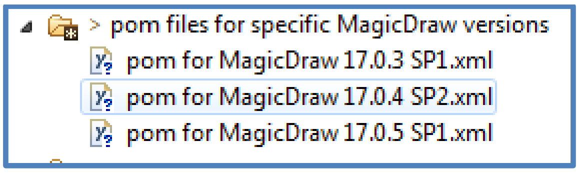
1. By default, the pom.xml file is set up for MagicDraw 17.0.3. **If you are using MagicDraw 17.0.3, you can skip this installation step**. If you are using a different MagicDraw version, such as 17.0.4 or 17.0.5, you will need to go through the next steps.
2. Each MagicDraw version comes with a different set of jars which need to be loaded on the Java classpath in order to use the MagicDraw API. The Maven ***pom.xml*** file refers to the jars which need to be on the Java classpath. In the folder named **pom files for specific MagicDraw versions**,

    a. **select** the pom file adequate to your version. For example, for MagicDraw 17.0.4 SP2, select the pom file named "*pom for MagicDraw 17.0.4 SP2*".

    b. Copy the file under the project root, so under the project named **org.eclipse.lyo.adapter.magicdraw**.

    c. *Then* **delete** the **pom.xml** file which was already under the project root,

    d. and **rename** the pom file you have just copied into the project root to **pom.xml**



pom files for specific MagicDraw versions

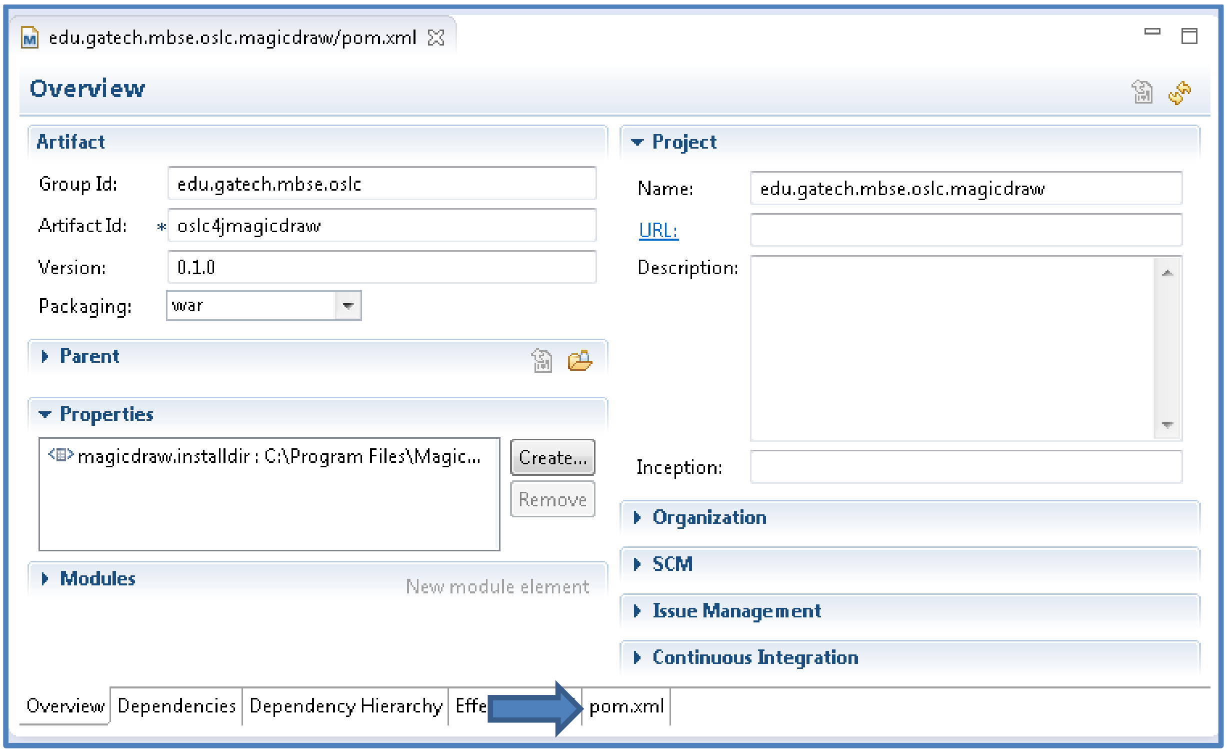
pom for MagicDraw 17.0.3 SP1.xml

pom for MagicDraw 17.0.4 SP2.xml

pom for MagicDraw 17.0.5 SP1.xml

5. Adding Proprietary MagicDraw Jars and documents to the project

1. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
2. Expand the **org.eclipse.lyo.adapter.magicdraw** project
3. Select and open the maven **pom.xml** file through double-click
4. The pom.xml file contains several tabs. By default, the overview tab will be displayed. The various available tabs are displayed at the bottom of the editor window as shown in the figure below. Click on the **pom.xml tab** of the pom.xml file as highlighted below.



edu.gatech.mbse.oslc.magicdraw/pom.xml

Overview

Artifact

Group Id: edu.gatech.mbse.oslc

Artifact Id: oslc4jmagicdraw

Version: 0.1.0

Packaging: war

Parent

Properties

magicdraw.installdir : C:\Program Files\Magic...

Modules

Project

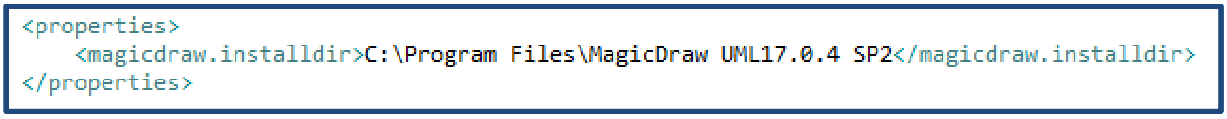
Name: edu.gatech.mbse.oslc.magicdraw

URL:

Description:

Inception:

5. In the pom.xml tab of the pom.xml file, specify the **location of the MagicDraw installation directory** in the **properties section** as highlighted below.



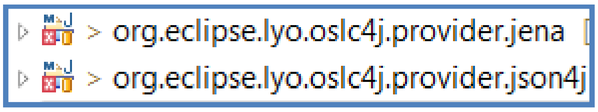
```
<properties>
    <magicdraw.installdir>C:\Program Files\MagicDraw UML17.0.4 SP2</magicdraw.installdir>
</properties>
```

6. Building the org.eclipse.lyo.adapter.magicdraw projects

1. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
2. Expand the **org.eclipse.lyo.adapter.magicdraw.ecore** project
3. Right click pom.xml -> Run As -> Maven clean
4. Right click pom.xml -> Run As -> Maven install
5. Expand the **org.eclipse.lyo.adapter.magicdraw.resources** project
6. Right click pom.xml -> Run As -> Maven clean
7. Right click pom.xml -> Run As -> Maven install
8. Expand the **org.eclipse.lyo.adapter.magicdraw** project
9. Right click pom.xml -> Run As -> Maven clean
10. Right click pom.xml -> Run As -> Maven install

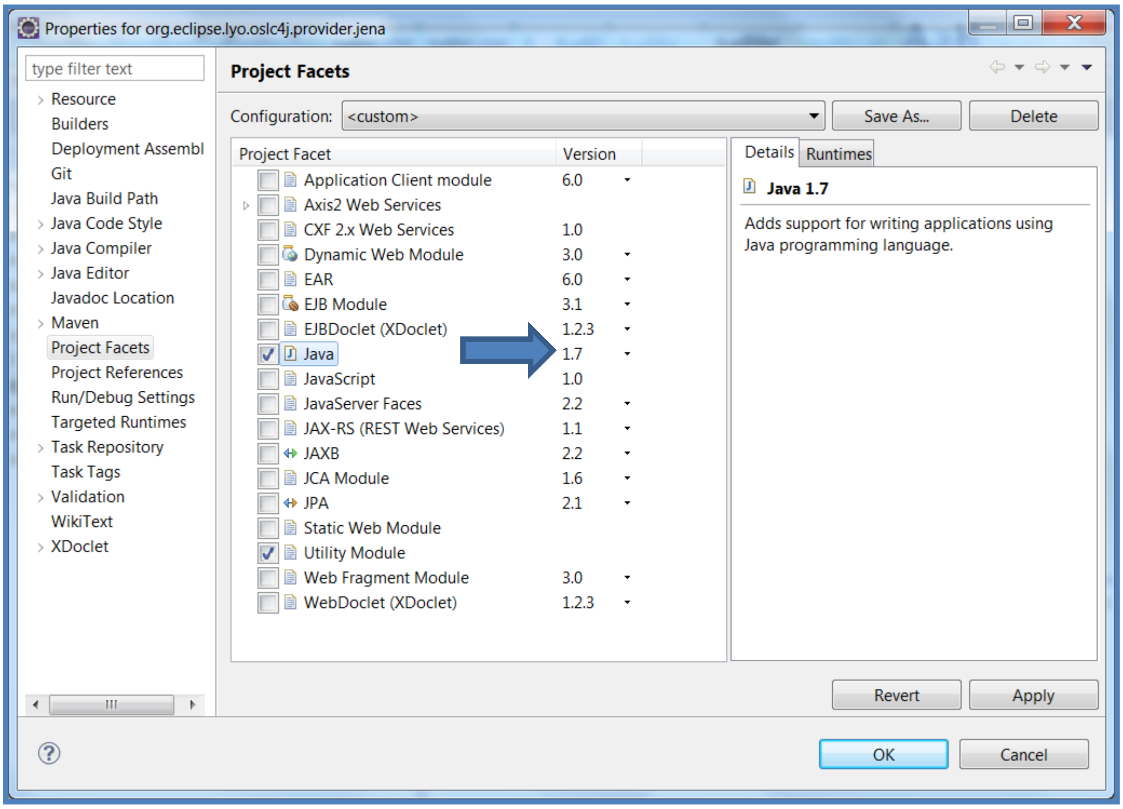
If there is no error mark next to any project, you can skip the next steps.

11. If there is a red error mark next to any project, select the project. Right-click->Maven->Update Project... and click OK
12. Make sure that the Eclipse projects displayed in the project explorer view do not contain any error icons displayed next to the project names as for example displayed below.



> org.eclipse.lyo.oslc4j.provider.jena
> org.eclipse.lyo.oslc4j.provider.json4j

If a project has an error icon, then select the project and open its properties view (Project->right click->Properties). Under the Projects Facet tab, make sure that 1.7 is selected as Java version as shown below.



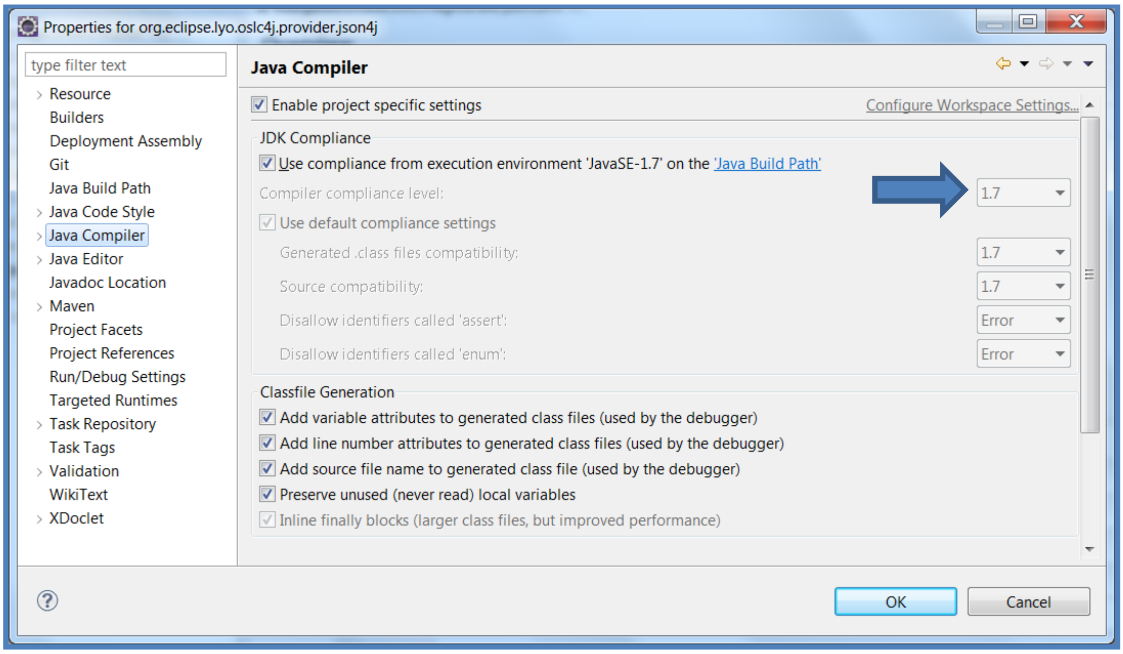
Configuration: <custom>

Project Facet	Version
☐ Application Client module	6.0
☐ Axis2 Web Services	
☐ CXF 2.x Web Services	1.0
☐ Dynamic Web Module	3.0
☐ EAR	6.0
☐ EJB Module	3.1
☐ EJBDoclet (XDoclet)	1.2.3
☑ Java	1.7
☐ JavaScript	1.0
☐ JavaServer Faces	2.2
☐ JAX-RS (REST Web Services)	1.1
☐ JAXB	2.2
☐ JCA Module	1.6
☐ JPA	2.1
☐ Static Web Module	
☑ Utility Module	
☐ Web Fragment Module	3.0
☐ WebDoclet (XDoclet)	1.2.3

Java 1.7

Adds support for writing applications using Java programming language.

If a project still shows an error, then change its **JDK compliance** to 1.7. Select the project, right-click -> Properties. Select Java Compiler and select **1.7** in the drop down menu next to the JDK compliance setting as highlighted below.



☑ Enable project specific settings

JDK Compliance

☑ Use compliance from execution environment 'JavaSE-1.7' on the 'Java Build Path'

Compiler compliance level: 1.7

☑ Use default compliance settings

Generated .class files compatibility: 1.7

Source compatibility: 1.7

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

☑ Inline finally blocks (larger class files, but improved performance)

7. Manual configuration

The OSLC MagicDraw adapter currently supports the retrieval of MagicDraw projects within a specific directory. The location of the directory containing the MagicDraw projects is currently hard coded in a configuration file. Several MagicDraw projects are already located in the ***MagicDraw Projects*** folder in the **org.eclipse.lyo.adapter.magicdraw** project.

1. Specify the **location of the folder containing MagicDraw models** which will be considered by the OSLC MagicDraw SysML adapter in the config.properties file under *org.eclipse.lyo.adapter.magicdraw/configuration*. As an example displayed below, the location of the folder containing MagicDraw models for the OSLC adapter is specified to **C:/Users/…/git/oslc4jmagicdraw/org.eclipse.lyo.adapter.magicdraw/Magicdraw Models/**

   Note: The file path can contain backslashes

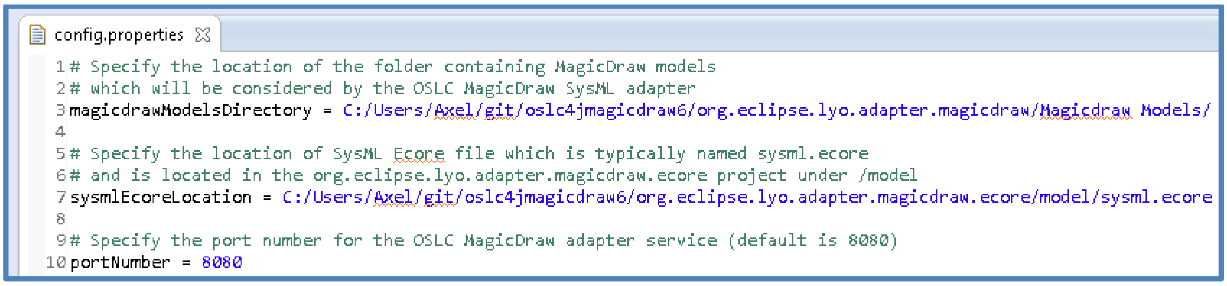
   Warning: Do not put quotes around the file path!

2. Specify the **location of SysML Ecore file** in the config.properties file under *org.eclipse.lyo.adapter.magicdraw/configuration*. The location of the SysML ecore file named sysml.ecore is in the org.eclipse.lyo.adapter.magicdraw.ecore project under /model/sysml.ecore. As an example displayed below, the location of the sysml.ecore file is specified to **C:/Users/…/git/oslc4jmagicdraw/org.eclipse.lyo.adapter.magicdraw/model/sysml.ecore**

   Note: The file path can contain backslashes

   Warning: Do not put quotes around the file path and add nothing at the end!

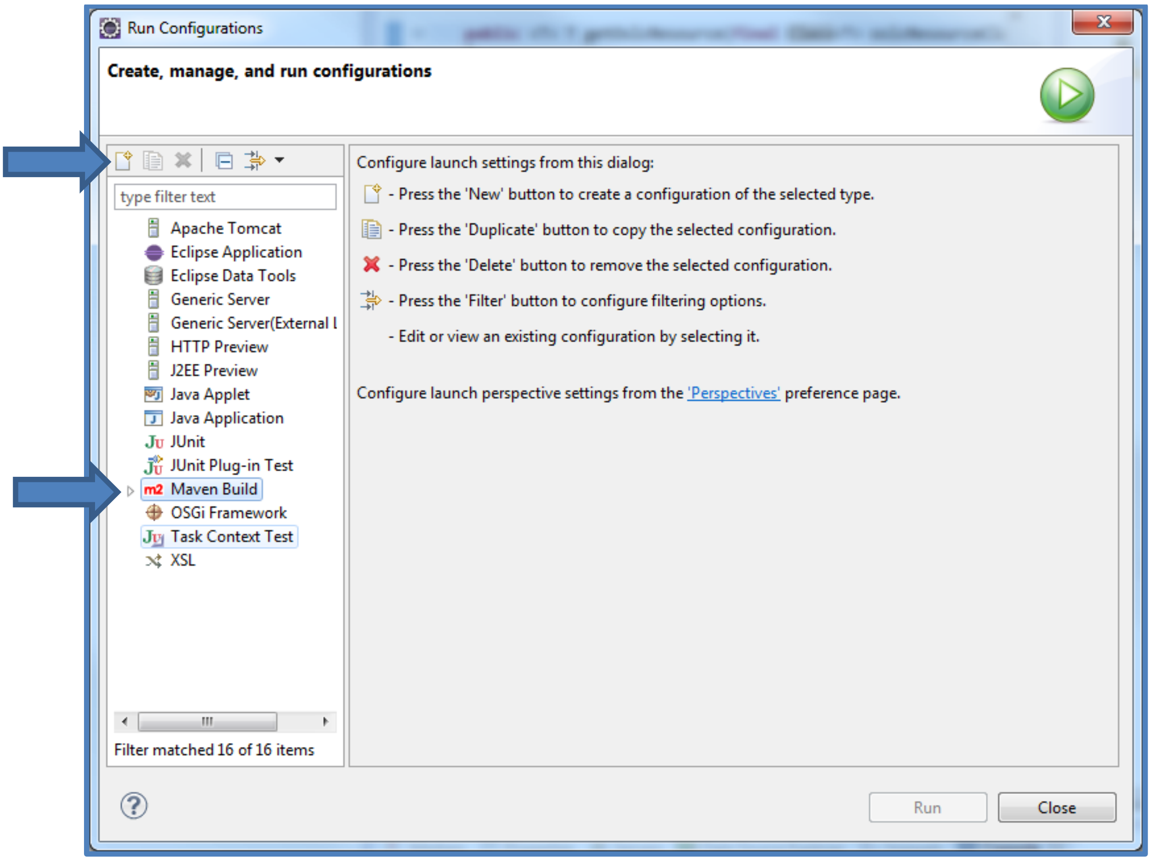
3. Specify the **port number** of the OSLC MagicDraw adapter service of in the config.properties file under *org.eclipse.lyo.adapter.magicdraw/configuration*. By default, port 8080 will be used. As an example displayed below, the port number is set to **8080**.



```
# Specify the location of the folder containing MagicDraw models
# which will be considered by the OSLC MagicDraw SysML adapter
magicdrawModelsDirectory = C:/Users/Axel/git/oslc4jmagicdraw6/org.eclipse.lyo.adapter.magicdraw/Magicdraw Models/

# Specify the location of SysML Ecore file which is typically named sysml.ecore
# and is located in the org.eclipse.lyo.adapter.magicdraw.ecore project under /model
sysmlEcoreLocation = C:/Users/Axel/git/oslc4jmagicdraw6/org.eclipse.lyo.adapter.magicdraw.ecore/model/sysml.ecore

# Specify the port number for the OSLC MagicDraw adapter service (default is 8080)
portNumber = 8080
```

8. Creating a launch configuration for org.eclipse.lyo.adapter.magicdraw

1. Open the launch configuration window (Run -> Run Configurations…)
2. Select the “Maven Build category” and click on “New Launch Configuration” (leftmost icon at the top left, right above the searh field)



Run Configurations

Create, manage, and run configurations

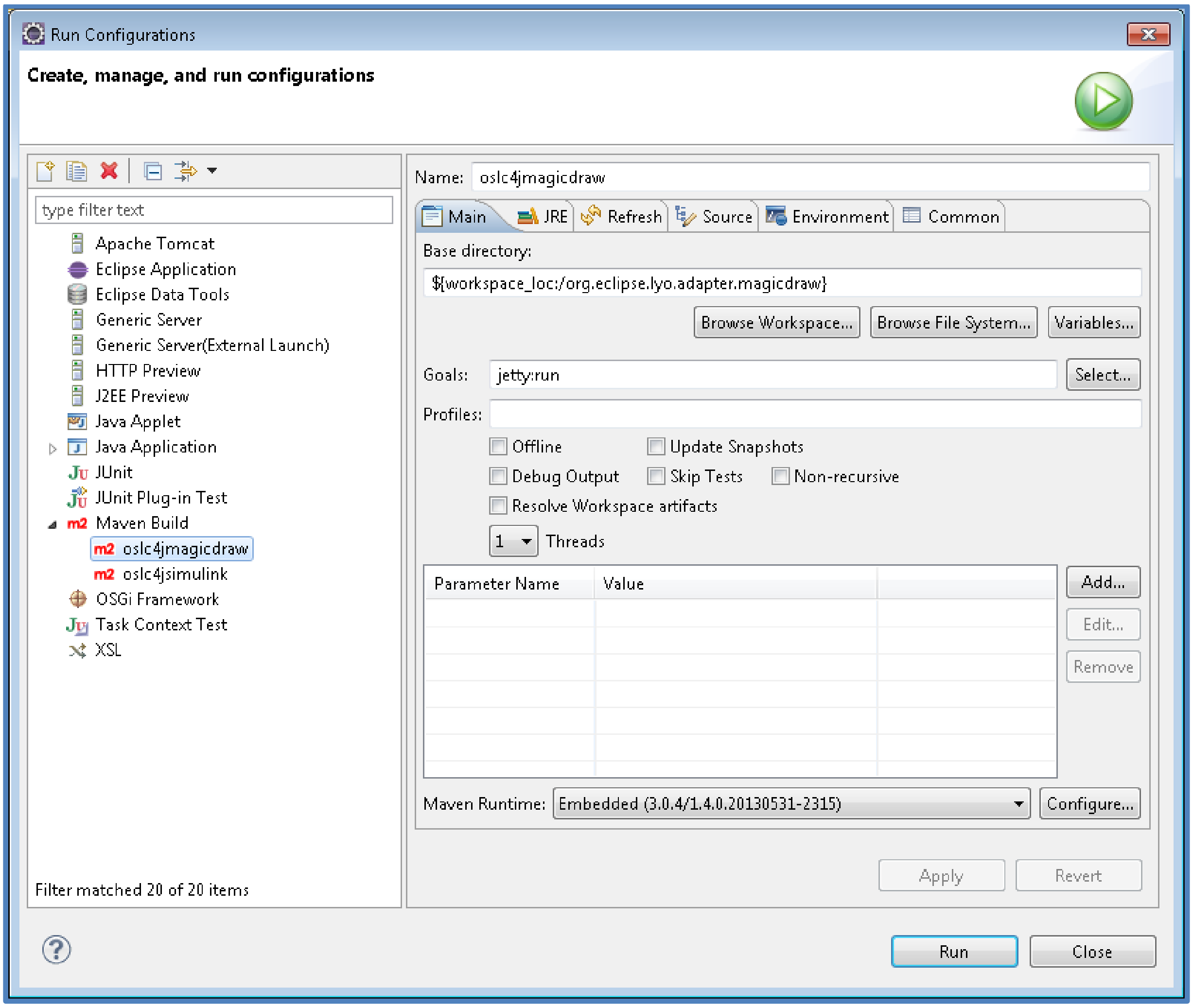
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Filter matched 16 of 16 items

3. In the launch configuration window, edit the name field and set it to **oslc4jmagicdraw**.
4. Under “Base Directory”, select Browse Workspace and select the **org.eclipse.lyo.adapter.magicdraw** project
5. In the “Goals” field, set **jetty:run.**
6. If the OSLC adapter for MagicDraw is supposed to run on a different port than 8080, in the “Goals” field, set **-Djetty.port=<port number> jetty:run**, such as: **-Djetty.port=9999 jetty:run**
Note: Make sure that the port number, if different than 8080, is also set in Step #7



Name: oslc4jmagicdraw

Base directory: ${workspace_loc:/org.eclipse.lyo.adapter.magicdraw}

Goals: jetty:run

Profiles:

□ Offline □ Update Snapshots

□ Debug Output □ Skip Tests □ Non-recursive

□ Resolve Workspace artifacts

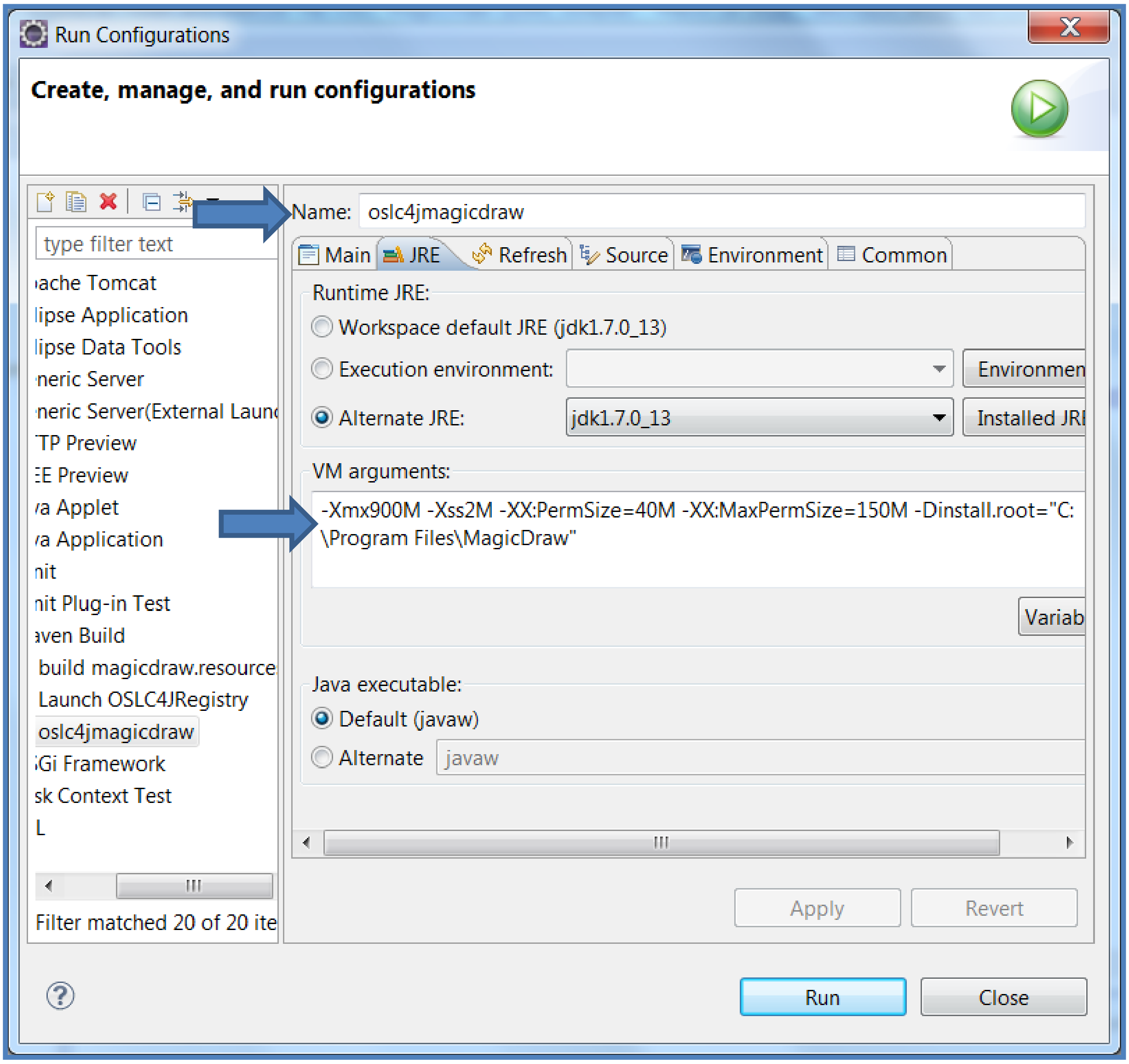
1 Threads

Maven Runtime: Embedded (3.0.4/1.4.0.20130531-2315)

7. In the JRE tab, under VM arguments, place:
   -Xmx900M -Xss2M -XX:PermSize=40M -XX:MaxPermSize=150M
   followed by
   -Dinstall.root="<path to MagicDraw installation directory>", for example
   -Dinstall.root="C:\Program Files\MagicDraw UML17.0.4 SP2" as shown below.

   Note: If you your machine has more than 4GB of main memory, you can adapt your JVM settings to use more memory using for example following settings (but make sure to while avoid an OutOfMemory error):
   -Xmx3674M -XX:PermSize=60M -XX:MaxPermSize=1350M

   Warning: If you are using a floating license, then you may need to add following JVM arguments based on your local settings:
   -DFL_SERVER_ADDRESS=flexnet_license_server_address
   -DFL_SERVER_PORT=license_server_port
   -DFL_EDITION=magicdraw_edition



Name: oslc4jmagicdraw

Runtime JRE:

○ Workspace default JRE (jdk1.7.0_13)

○ Execution environment:

● Alternate JRE: jdk1.7.0_13

VM arguments:

```
-Xmx900M -Xss2M -XX:PermSize=40M -XX:MaxPermSize=150M -Dinstall.root="C:
\Program Files\MagicDraw"
```

Java executable:

● Default (javaw)

○ Alternate javaw

9. Installing the Chrome/Firefox Postman plugin (or any REST client)

1. For Google Chrome, add the Postman REST client to your browser: https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm?hl=en
2. And the Postman launcher: https://chrome.google.com/webstore/detail/postman-launcher/igofndmniooofoabmmpfonmdnhgchoka?hl=en

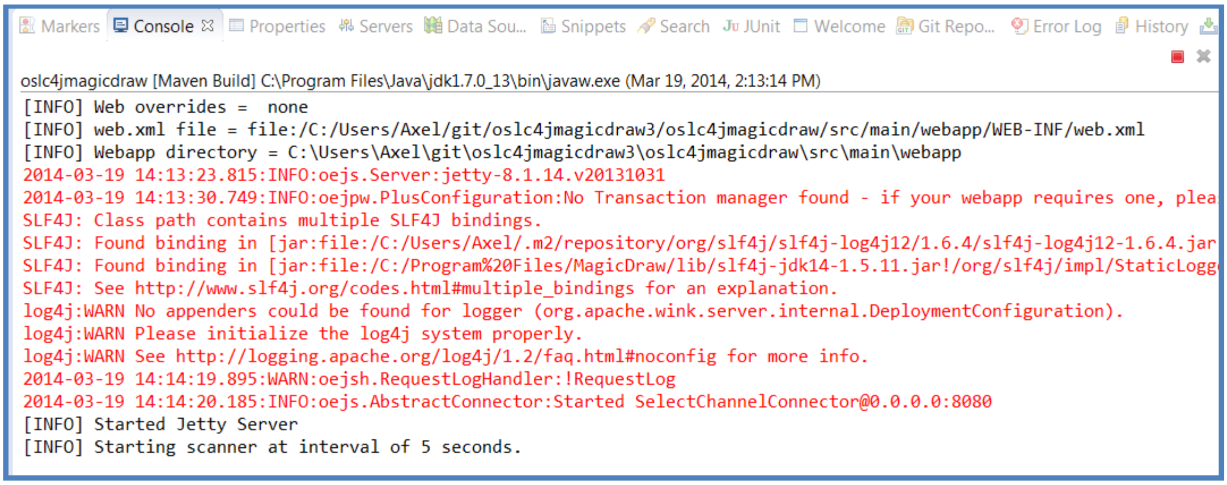
10. Launching org.eclipse.lyo.adapter.magicdraw (OSLC MagicDraw Adapter)

Select the oslc4jmagicdraw launch configuration (Run -> Run Configurations… and select in the Maven build category the launch configuration named **oslc4jmagicdraw**) and click Run.

Note: The Jetty server will launch. This will take a few minutes because it has to load many jars (especially all the MagicDraw jars).

In the console window, several logging related exceptions will appear (SLF4J and log4j). This is not critical.

The OSLC4 MagicDraw adapter is running and following statements can be seen in the Eclipse Console windows displayed below.



oslc4jmagicdraw [Maven Build] C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Mar 19, 2014, 2:13:14 PM)

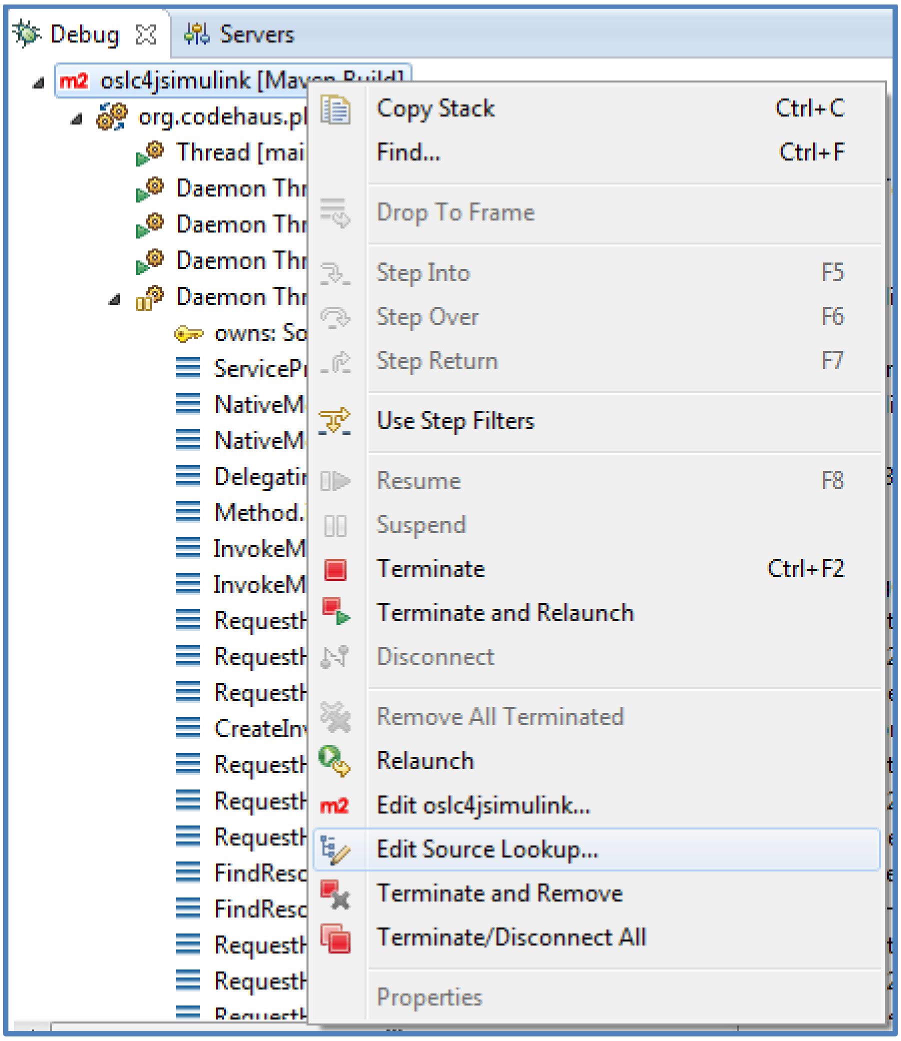
```
[INFO] Web overrides =  none
[INFO] web.xml file = file:/C:/Users/Axel/git/oslc4jmagicdraw3/oslc4jmagicdraw/src/main/webapp/WEB-INF/web.xml
[INFO] Webapp directory = C:\Users\Axel\git\oslc4jmagicdraw3\oslc4jmagicdraw\src\main\webapp
2014-03-19 14:13:23.815:INFO:oejs.Server:jetty-8.1.14.v20131031
2014-03-19 14:13:30.749:INFO:oejpw.PlusConfiguration:No Transaction manager found - if your webapp requires one, plea
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/Users/Axel/.m2/repository/org/slf4j/slf4j-log4j12/1.6.4/slf4j-log4j12-1.6.4.jar
SLF4J: Found binding in [jar:file:/C:/Program%20Files/MagicDraw/lib/slf4j-jdk14-1.5.11.jar!/org/slf4j/impl/StaticLogg
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
log4j:WARN No appenders could be found for logger (org.apache.wink.server.internal.DeploymentConfiguration).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2014-03-19 14:14:19.895:WARN:oejsh.RequestLogHandler:!RequestLog
2014-03-19 14:14:20.185:INFO:oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 5 seconds.
```

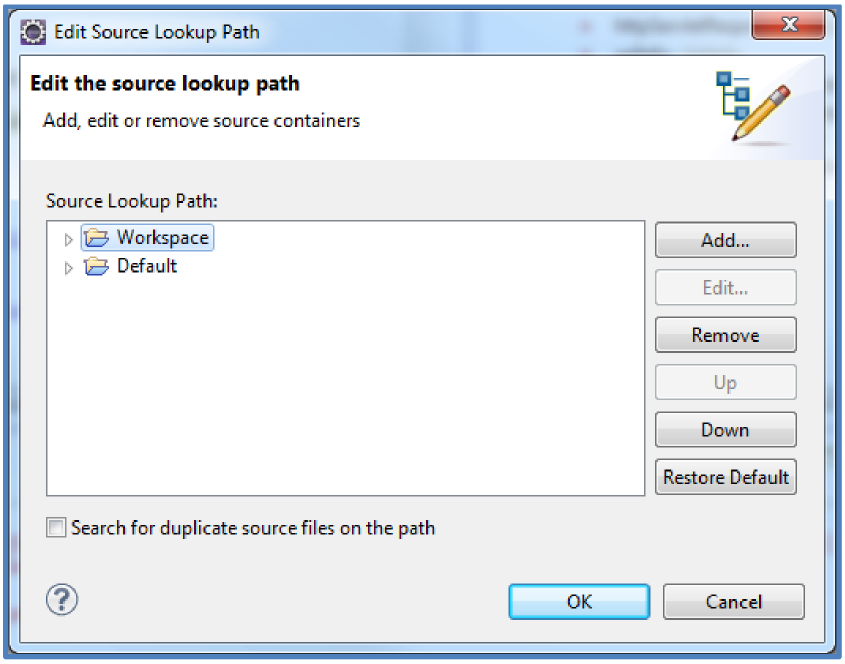
Warning: If the Jetty server does not launch successfully at the **first attempt**, try again. It has been observed that the Jetty server does not launch successfully at the first attempt, possibly due to a time-out error after the time-consuming download of Jetty-related artifacts by Maven.

Warning: If the Jetty server fails to launch due to a **java.net.BindException**, a different port for the OSLC MagicDraw adapter needs to be used since there is a conflict with another service using the same port. By default, Jetty uses port 8080. A **java.net.BindException** means that a different service is already using this port. Go back to Steps #7 and #8 to change the port number.

Note: If there is a problem with launching MagicDraw through the OSLC MagicDraw adapter, check the MagicDraw log file which can be found by choosing in the MagicDraw menu bar Help->About MagicDraw->Environment and clicking on the link to the log file.

Note: If you launch the Maven launch configuration (OSLC MagicDraw adapter) in **debug mode**, and do not see the Java code when the application hits a breakpoint, then you need to add the Eclipse workspace to the source lookup path. In the Debug view, right click on the running thread (in threads tab), or on the application as shown in the example below and select **Edit Source Lookup**, and add the workspace. Re-launch the Maven launch configuration and the code should be visible in the editor when the application hits a breakpoint.

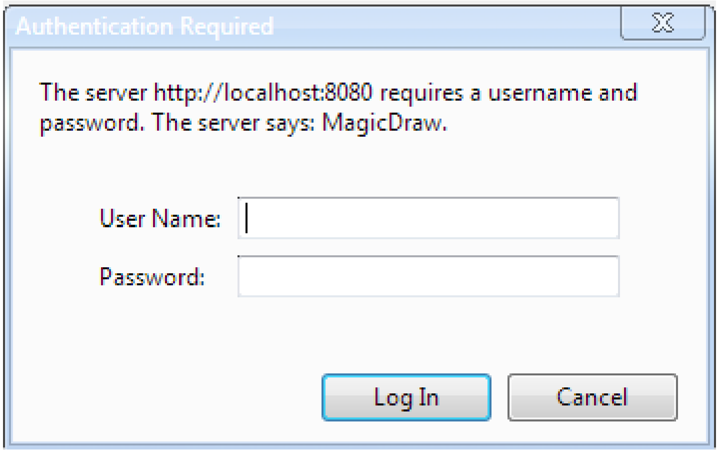



Edit Source Lookup Path
Edit the source lookup path
Add, edit or remove source containers
Source Lookup Path:
Workspace
Default
Add...
Edit...
Remove
Up
Down
Restore Default
□ Search for duplicate source files on the path
OK
Cancel

11. Testing the OSLC MagicDraw Adapter

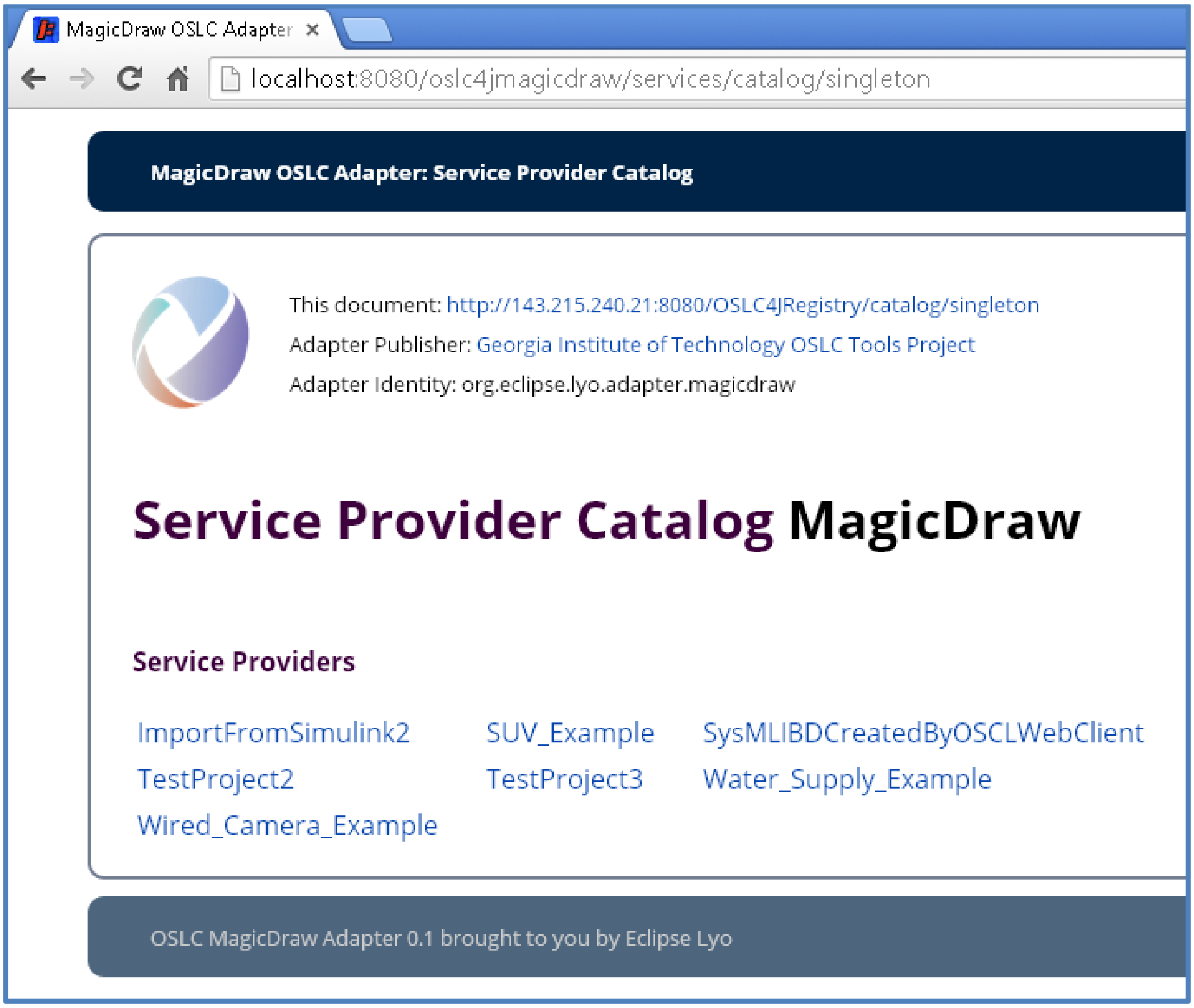
Testing the retrieval of OSLC resources in HTML

1. Launch Google Chrome
2. In the URL field, type for test purposes: http://localhost:8080/oslc4jmagicdraw/services/catalog/singleton. This will send a HTTP GET request to retrieve the HTML representation of the MagicDraw Service Provider Catalog
3. You may need to authenticate yourself. If you do, just type in any user name and password. If you do not want the browser to save the credentials, use the Google Chrome incognito window.


Authentication Required

The server http://localhost:8080 requires a username and password. The server says: MagicDraw.

4. You will then see an HTML page showing you the list of Service Providers. You can browse from the Service Providers (e.g. for TestProject2) to the Services and ultimately to the SysML resources.



MagicDraw OSLC Adapter: Service Provider Catalog

This document: http://143.215.240.21:8080/OSLC4JRegistry/catalog/singleton

Adapter Publisher: Georgia Institute of Technology OSLC Tools Project

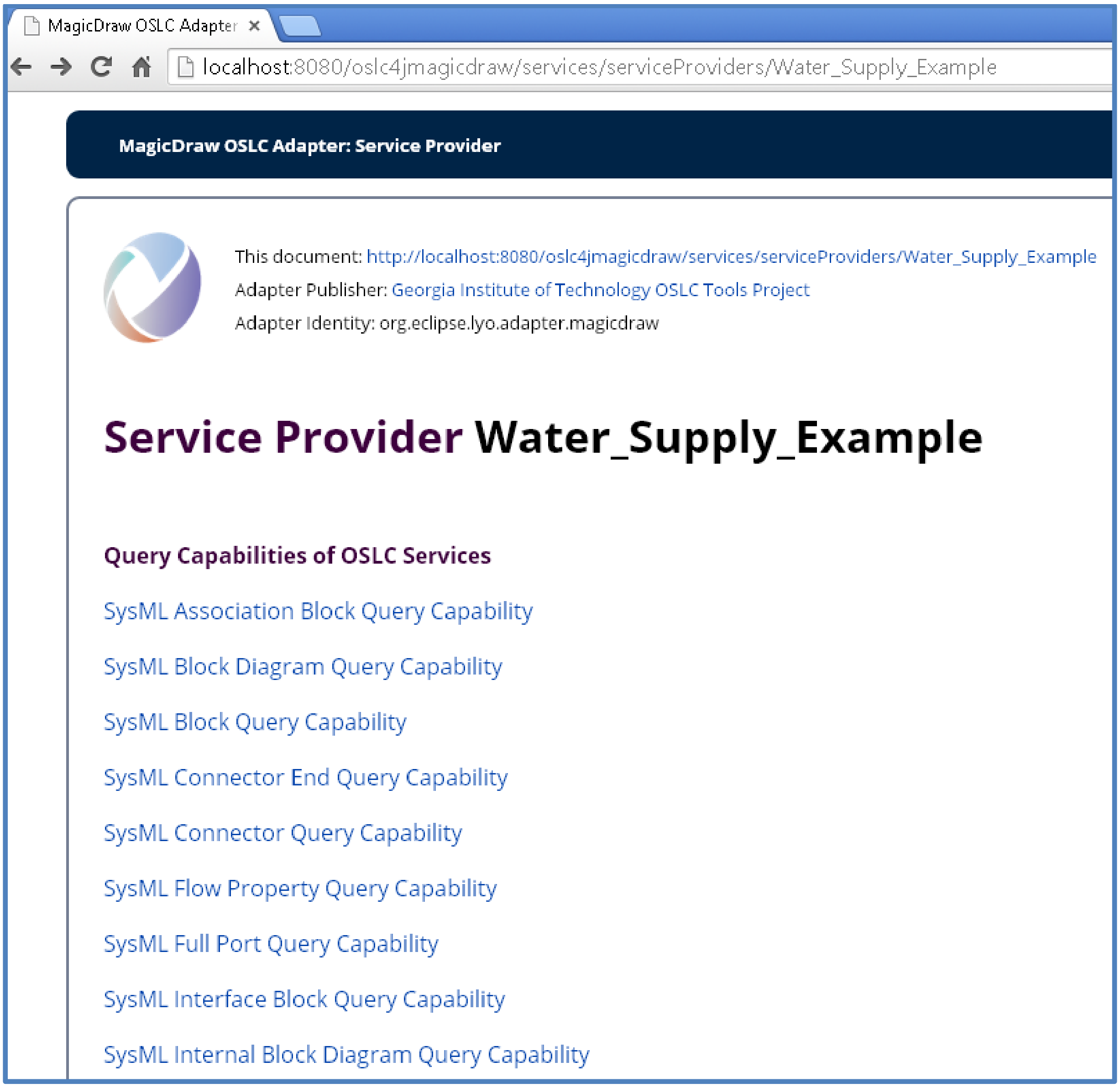
Adapter Identity: org.eclipse.lyo.adapter.magicdraw

Service Provider Catalog MagicDraw

Service Providers

ImportFromSimulink2
TestProject2
Wired_Camera_Example
SUV_Example
TestProject3
SysMLIBDCreatedByOSCLWebClient
Water_Supply_Example

OSLC MagicDraw Adapter 0.1 brought to you by Eclipse Lyo



MagicDraw OSLC Adapter
localhost:8080/oslc4jmagicdraw/services/serviceProviders/Water_Supply_Example

MagicDraw OSLC Adapter: Service Provider

This document: http://localhost:8080/oslc4jmagicdraw/services/serviceProviders/Water_Supply_Example

Adapter Publisher: Georgia Institute of Technology OSLC Tools Project

Adapter Identity: org.eclipse.lyo.adapter.magicdraw

Service Provider Water_Supply_Example

Query Capabilities of OSLC Services

SysML Association Block Query Capability

SysML Block Diagram Query Capability

SysML Block Query Capability

SysML Connector End Query Capability

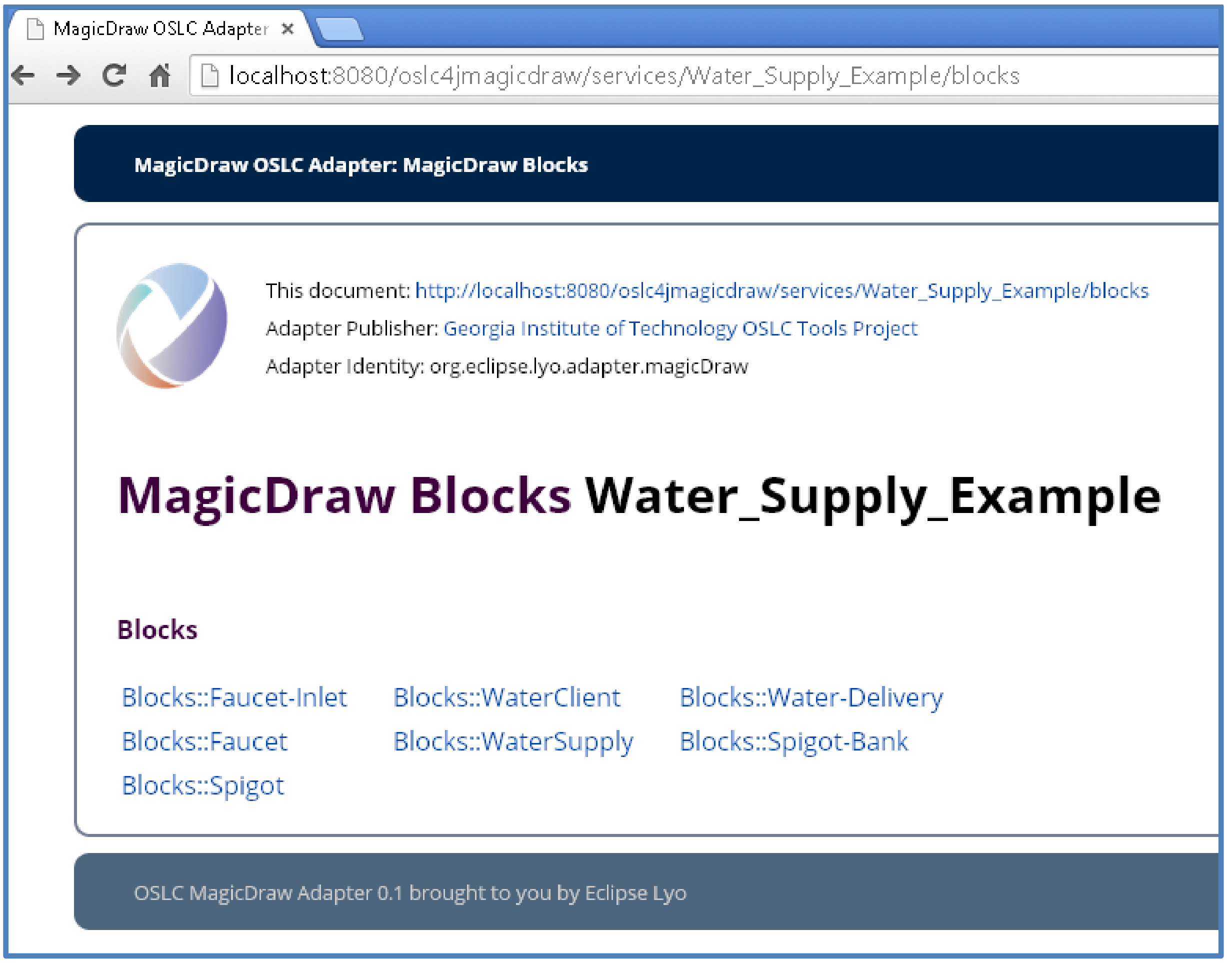
SysML Connector Query Capability

SysML Flow Property Query Capability

SysML Full Port Query Capability

SysML Interface Block Query Capability

SysML Internal Block Diagram Query Capability



MagicDraw OSLC Adapter
localhost:8080/oslc4jmagicdraw/services/Water_Supply_Example/blocks

MagicDraw OSLC Adapter: MagicDraw Blocks

This document: http://localhost:8080/oslc4jmagicdraw/services/Water_Supply_Example/blocks

Adapter Publisher: Georgia Institute of Technology OSLC Tools Project

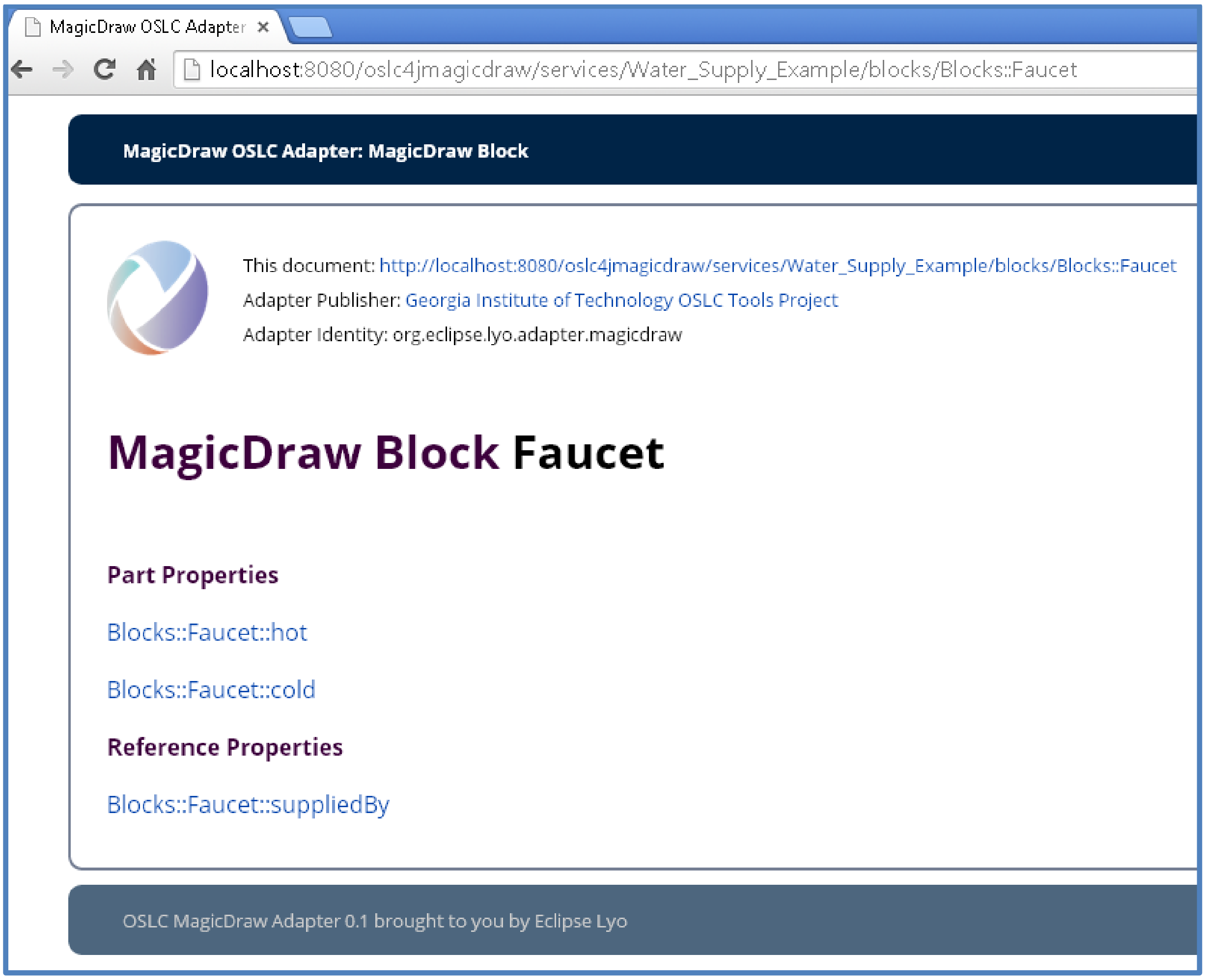
Adapter Identity: org.eclipse.lyo.adapter.magicDraw

MagicDraw Blocks Water_Supply_Example

Blocks

Blocks::Faucet-Inlet
Blocks::Faucet
Blocks::Spigot
Blocks::WaterClient
Blocks::WaterSupply
Blocks::Water-Delivery
Blocks::Spigot-Bank

OSLC MagicDraw Adapter 0.1 brought to you by Eclipse Lyo


MagicDraw OSLC Adapter
localhost:8080/oslc4jmagicdraw/services/Water_Supply_Example/blocks/Blocks::Faucet

MagicDraw OSLC Adapter: MagicDraw Block

This document: http://localhost:8080/oslc4jmagicdraw/services/Water_Supply_Example/blocks/Blocks::Faucet

Adapter Publisher: Georgia Institute of Technology OSLC Tools Project

Adapter Identity: org.eclipse.lyo.adapter.magicdraw

MagicDraw Block Faucet

Part Properties

Blocks::Faucet::hot

Blocks::Faucet::cold

Reference Properties

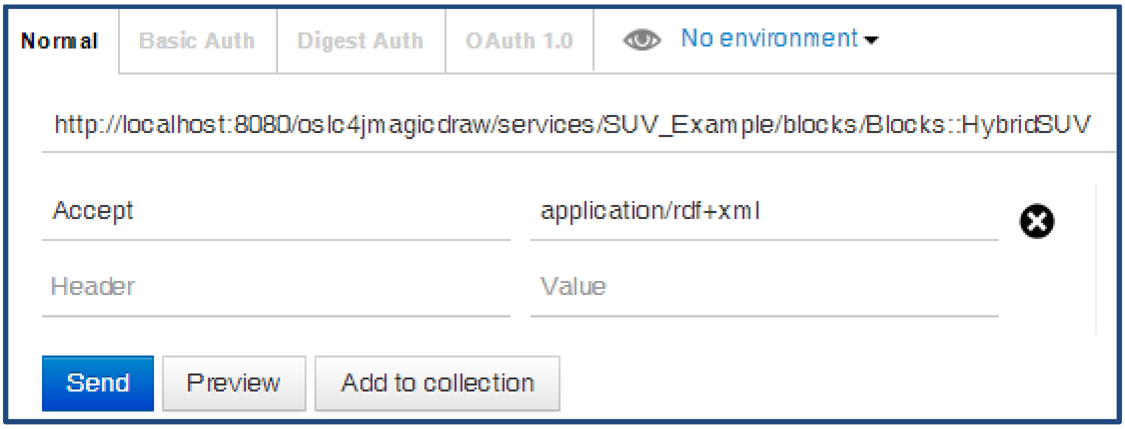
Blocks::Faucet::suppliedBy

OSLC MagicDraw Adapter 0.1 brought to you by Eclipse Lyo

Testing the retrieval of OSLC resources in RDF

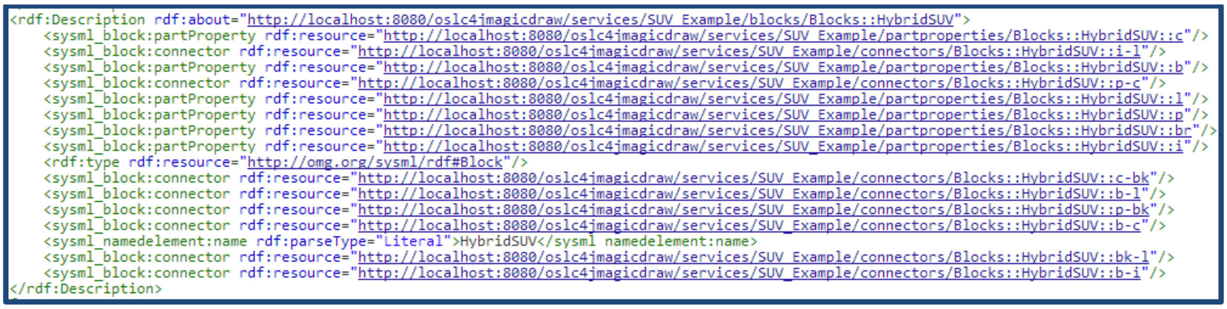
1. Click on the Postman icon at the top right of the Chrome browser . A new tab will open.
2. In the URL field, type for test purposes: http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/blocks/Blocks::HybridSUV?Accept=application/rdf+xml. This will send a HTTP GET request to retrieve the RDF/XML representation of the SysML block named “HybridSUV”.

The Postman REST client will display the RDF/XML representation of the SysML block named “Faucet”. Other HTTP requests to retrieve other SysML elements can be sent.



http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/blocks/Blocks::HybridSUV

Accept: application/rdf+xml

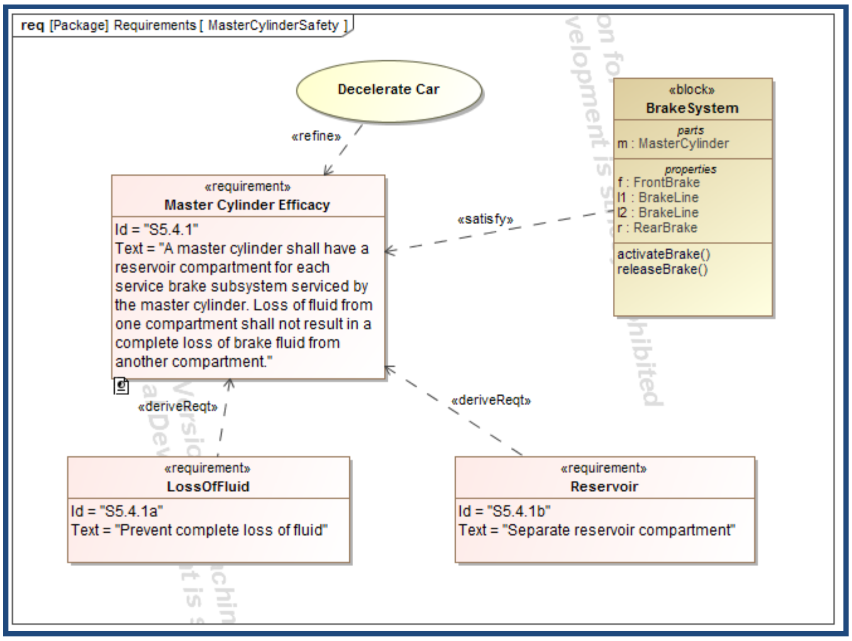


```
<rdf:Description rdf:about="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/blocks/Blocks::HybridSUV">
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::c"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::i-l"/>
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::b"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::p-c"/>
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::l"/>
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::p"/>
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::br"/>
    <sysml_block:partProperty rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/partproperties/Blocks::HybridSUV::i"/>
    <rdf:type rdf:resource="http://omg.org/sysml/rdf#Block"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::c-bk"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::b-l"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::p-bk"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::b-c"/>
    <sysml_namedelement:name rdf:parseType="Literal">HybridSUV</sysml_namedelement:name>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::bk-l"/>
    <sysml_block:connector rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/SUV_Example/connectors/Blocks::HybridSUV::b-i"/>
</rdf:Description>
```

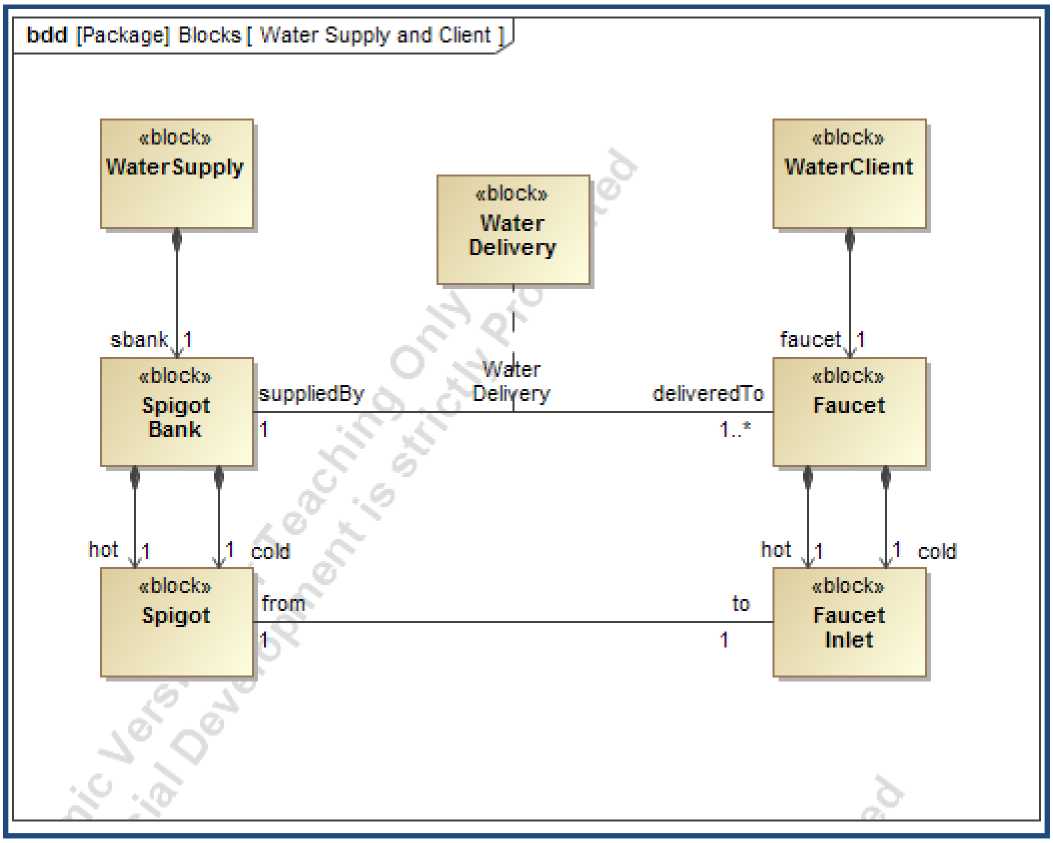
12. Testing the OSLC MagicDraw Adapter through example MagicDraw models

The org.eclipse.lyo.adapter.magicdraw project contains example MagicDraw models containing different types of SysML elements. The example models are located in the folder named “MagicDraw Models”.

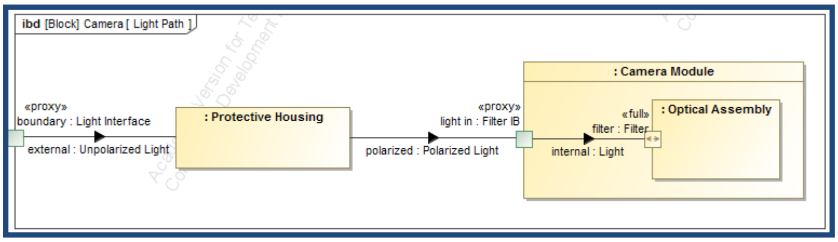
SUV Example contains requirements and relationships between requirements and different SysML elements.

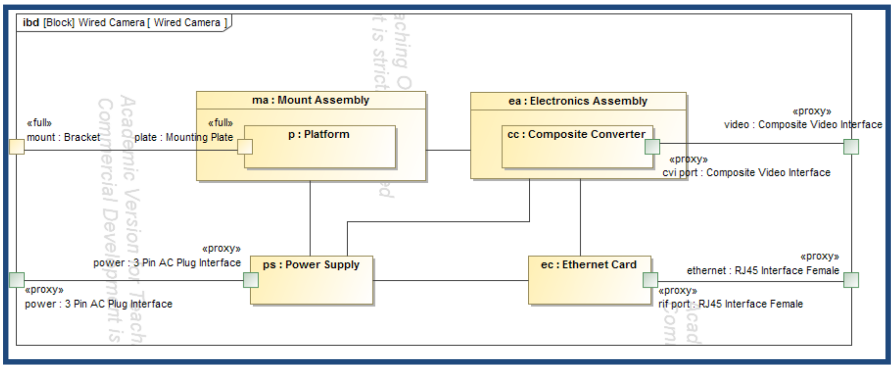

req [Package] Requirements [MasterCylinderSafety]
Decelerate Car
«refine»
«requirement»
Master Cylinder Efficacy
Id = "S5.4.1"
Text = "A master cylinder shall have a reservoir compartment for each service brake subsystem serviced by the master cylinder. Loss of fluid from one compartment shall not result in a complete loss of brake fluid from another compartment."
«satisfy»
«block»
BrakeSystem
parts
m : MasterCylinder
properties
f : FrontBrake
l1 : BrakeLine
l2 : BrakeLine
r : RearBrake
activateBrake()
releaseBrake()
«deriveReqt»
«deriveReqt»
«requirement»
LossOfFluid
Id = "S5.4.1a"
Text = "Prevent complete loss of fluid"
«requirement»
Reservoir
Id = "S5.4.1b"
Text = "Separate reservoir compartment"

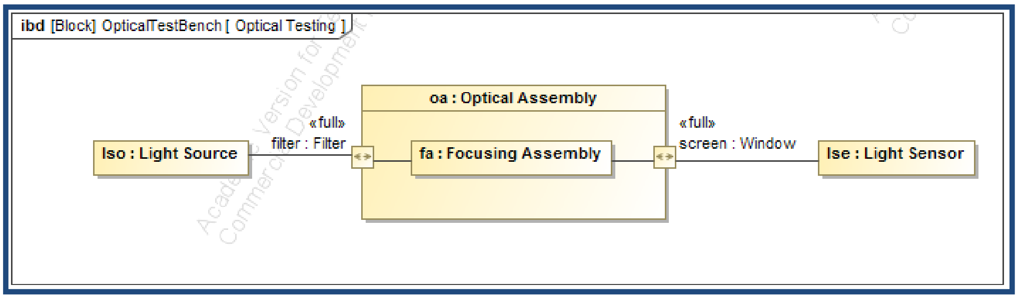
Water Supply Example contains blocks, associations, and associationblocks.



Wired Camera Example contains parts, connectors, ports, full ports, proxy ports, item flows




ibd [Block] Wired Camera [Wired Camera]
«full»
mount : Bracket
«full»
plate : Mounting Plate
ma : Mount Assembly
p : Platform
ea : Electronics Assembly
cc : Composite Converter
«proxy»
video : Composite Video Interface
«proxy»
cvi port : Composite Video Interface
«proxy»
power : 3 Pin AC Plug Interface
ps : Power Supply
«proxy»
power : 3 Pin AC Plug Interface
ec : Ethernet Card
«proxy»
ethernet : RJ45 Interface Female
«proxy»
rif port : RJ45 Interface Female

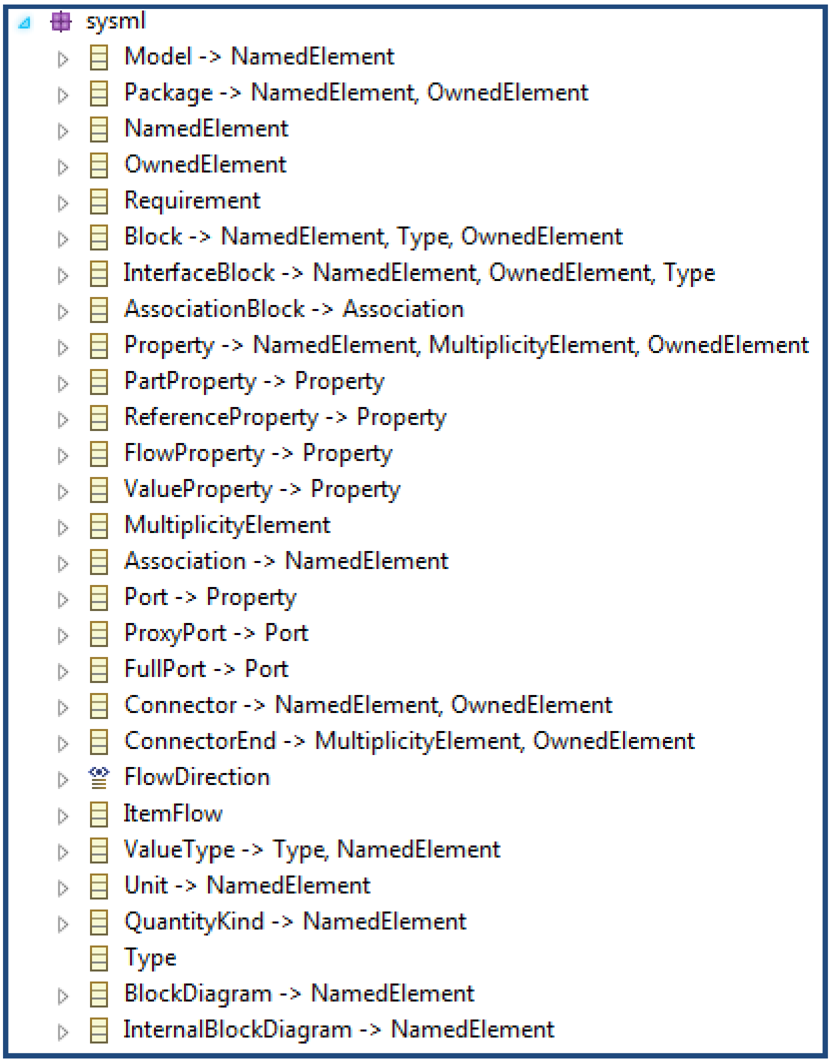


13. Getting an overview of SysML concepts supported by the OSLC MagicDraw adapter

All the concepts supported by the OSLC MagicDraw adapter have been defined in an Ecore metamodel named **sysml.ecore**, located in the project named *org.eclipse.lyo.adapter.magicdraw.ecore*, under the folder named *model*.

The ecore metamodel is displayed in tree format below. It shows the list of SysML concepts which are supported by the OSLC MagicDraw SysML adapter.

Not all concepts are supported equally. As an example, the adapter has an OSLC service to create blocks, but there is no service for creating interface block diagrams. In addition, abstract SysML concepts such as NamedElement, which are reused across several SysML concepts, are not supported by the OSLC MagicDraw SysML adapter.

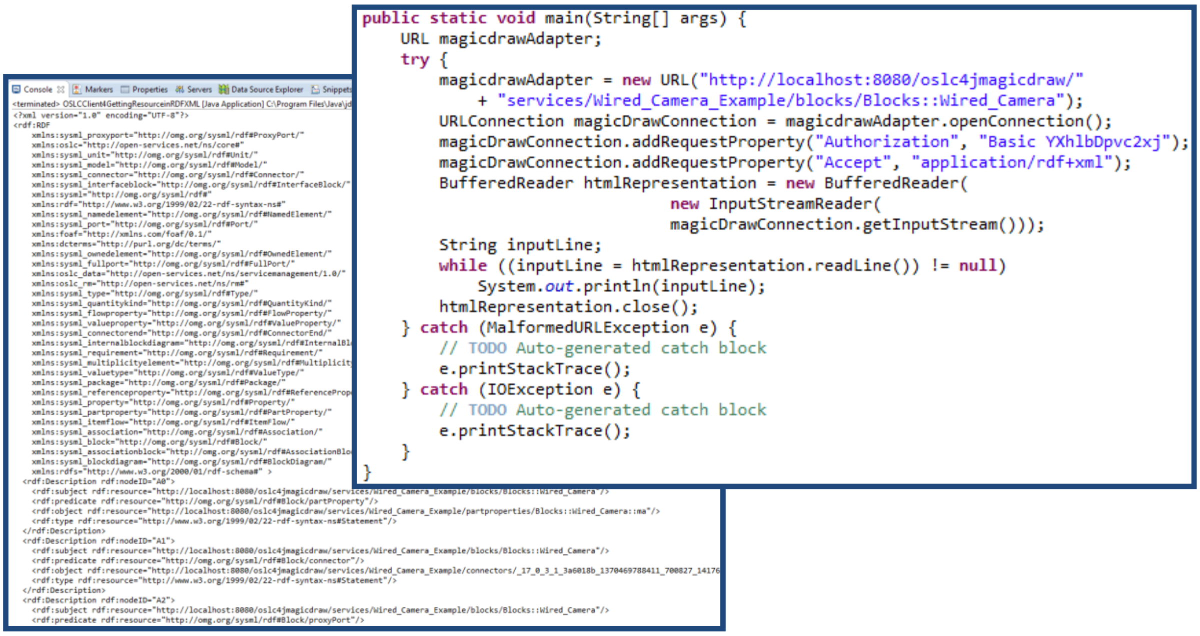


sysml
- Model -> NamedElement
- Package -> NamedElement, OwnedElement
- NamedElement
- OwnedElement
- Requirement
- Block -> NamedElement, Type, OwnedElement
- InterfaceBlock -> NamedElement, OwnedElement, Type
- AssociationBlock -> Association
- Property -> NamedElement, MultiplicityElement, OwnedElement
- PartProperty -> Property
- ReferenceProperty -> Property
- FlowProperty -> Property
- ValueProperty -> Property
- MultiplicityElement
- Association -> NamedElement
- Port -> Property
- ProxyPort -> Port
- FullPort -> Port
- Connector -> NamedElement, OwnedElement
- ConnectorEnd -> MultiplicityElement, OwnedElement
- FlowDirection
- ItemFlow
- ValueType -> Type, NamedElement
- Unit -> NamedElement
- QuantityKind -> NamedElement
- Type
- BlockDiagram -> NamedElement
- InternalBlockDiagram -> NamedElement

14. Accessing SysML elements as OSLC resources using Java clients

OSLC resources and services can be accessed through and the browser and any client which supports HTTP. Multiple Java-based clients have been developed to access OSLC resources and services programmatically. The clients are available as Java applications in the *org.eclipse.lyo.adapter.magicdraw.clients* package.

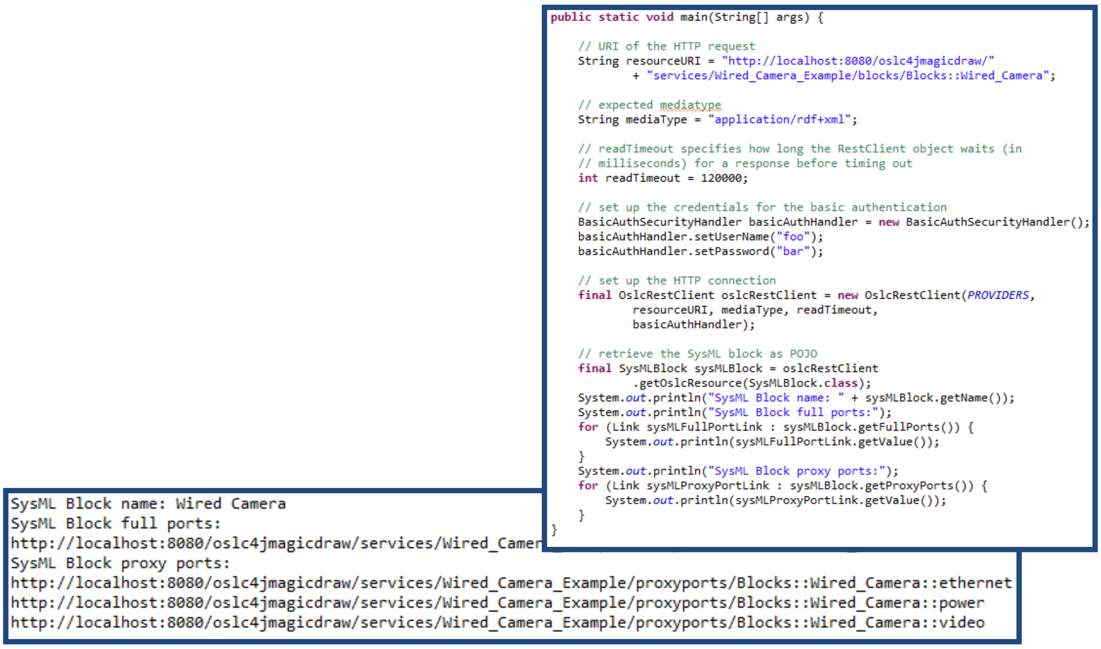
HTTPClient4GettingResourceRepresentation.java: This Java client retrieves an OSLC resource and prints into the console the representation of the OSLC resource.



```java
public static void main(String[] args) {
    URL magicdrawAdapter;
    try {
        magicdrawAdapter = new URL("http://localhost:8080/oslc4jmagicdraw/"
            + "services/Wired_Camera_Example/blocks/Blocks::Wired_Camera");
        URLConnection magicDrawConnection = magicdrawAdapter.openConnection();
        magicDrawConnection.addRequestProperty("Authorization", "Basic YXhlbDpvc2xj");
        magicDrawConnection.addRequestProperty("Accept", "application/rdf+xml");
        BufferedReader htmlRepresentation = new BufferedReader(
                                new InputStreamReader(
                                magicDrawConnection.getInputStream()));
        String inputLine;
        while ((inputLine = htmlRepresentation.readLine()) != null)
            System.out.println(inputLine);
        htmlRepresentation.close();
    } catch (MalformedURLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
<terminated> OSLCClient4GettingResourceinRDFXML [Java Application] C:\Program Files\Java\jd
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:sysml_proxyport="http://omg.org/sysml/rdf#ProxyPort/"
    xmlns:oslc="http://open-services.net/ns/core#"
    xmlns:sysml_unit="http://omg.org/sysml/rdf#Unit/"
    xmlns:sysml_model="http://omg.org/sysml/rdf#Model/"
    xmlns:sysml_connector="http://omg.org/sysml/rdf#Connector/"
    xmlns:sysml_interfaceblock="http://omg.org/sysml/rdf#InterfaceBlock/"
    xmlns:sysml="http://omg.org/sysml/rdf#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:sysml_namedelement="http://omg.org/sysml/rdf#NamedElement/"
    xmlns:sysml_port="http://omg.org/sysml/rdf#Port/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:sysml_ownedelement="http://omg.org/sysml/rdf#OwnedElement/"
    xmlns:sysml_fullport="http://omg.org/sysml/rdf#FullPort/"
    xmlns:oslc_data="http://open-services.net/ns/servicemanagement/1.0/"
    xmlns:oslc_rm="http://open-services.net/ns/rm#"
    xmlns:sysml_type="http://omg.org/sysml/rdf#Type/"
    xmlns:sysml_quantitykind="http://omg.org/sysml/rdf#QuantityKind/"
    xmlns:sysml_flowproperty="http://omg.org/sysml/rdf#FlowProperty/"
    xmlns:sysml_valueproperty="http://omg.org/sysml/rdf#ValueProperty/"
    xmlns:sysml_connectorend="http://omg.org/sysml/rdf#ConnectorEnd/"
    xmlns:sysml_internalblockdiagram="http://omg.org/sysml/rdf#InternalBl
    xmlns:sysml_requirement="http://omg.org/sysml/rdf#Requirement/"
    xmlns:sysml_multiplicityelement="http://omg.org/sysml/rdf#Multiplicit
    xmlns:sysml_valuetype="http://omg.org/sysml/rdf#ValueType/"
    xmlns:sysml_package="http://omg.org/sysml/rdf#Package/"
    xmlns:sysml_referenceproperty="http://omg.org/sysml/rdf#ReferenceProp
    xmlns:sysml_property="http://omg.org/sysml/rdf#Property/"
    xmlns:sysml_partproperty="http://omg.org/sysml/rdf#PartProperty/"
    xmlns:sysml_itemflow="http://omg.org/sysml/rdf#ItemFlow/"
    xmlns:sysml_association="http://omg.org/sysml/rdf#Association/"
    xmlns:sysml_block="http://omg.org/sysml/rdf#Block/"
    xmlns:sysml_associationblock="http://omg.org/sysml/rdf#AssociationBlo
    xmlns:sysml_blockdiagram="http://omg.org/sysml/rdf#BlockDiagram/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:nodeID="A0">
    <rdf:subject rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/blocks/Blocks::Wired_Camera"/>
    <rdf:predicate rdf:resource="http://omg.org/sysml/rdf#Block/partProperty"/>
    <rdf:object rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/partproperties/Blocks::Wired_Camera::ma"/>
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A1">
    <rdf:subject rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/blocks/Blocks::Wired_Camera"/>
    <rdf:predicate rdf:resource="http://omg.org/sysml/rdf#Block/connector"/>
    <rdf:object rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/connectors/_17_0_3_1_3a6018b_1370469788411_700827_14176
    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="A2">
    <rdf:subject rdf:resource="http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/blocks/Blocks::Wired_Camera"/>
    <rdf:predicate rdf:resource="http://omg.org/sysml/rdf#Block/proxyPort"/>
```

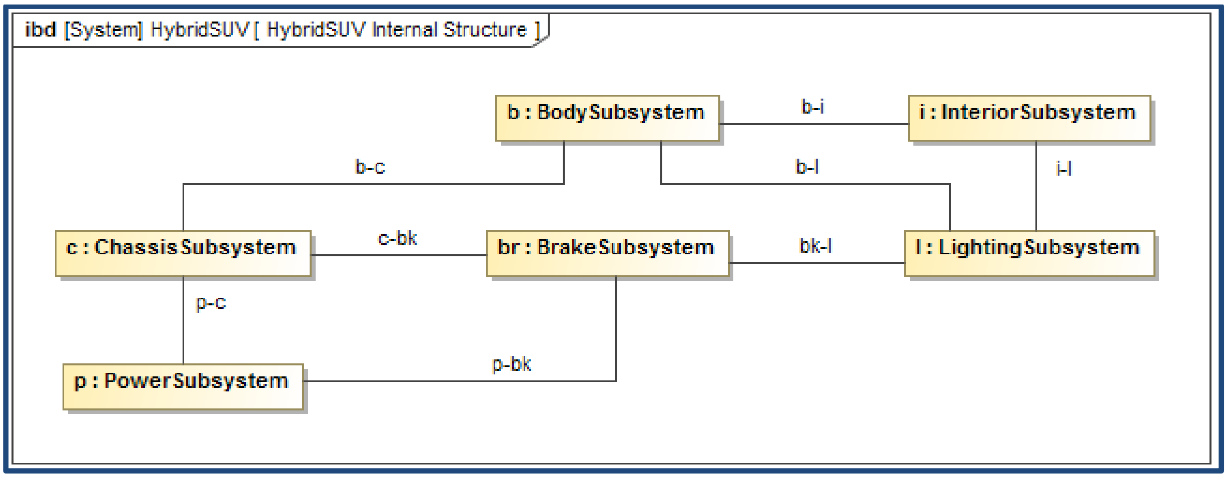
OSLCClient4GettingSysMLBlock.java: This client retrieves an OSLC resource describing a SysML block as a POJO and uses the OSLC4J library to programmatically retrieve the properties of the SysML block.



```java
public static void main(String[] args) {

    // URI of the HTTP request
    String resourceURI = "http://localhost:8080/oslc4jmagicdraw/"
            + "services/Wired_Camera_Example/blocks/Blocks::Wired_Camera";

    // expected mediatype
    String mediaType = "application/rdf+xml";

    // readTimeout specifies how long the RestClient object waits (in
    // milliseconds) for a response before timing out
    int readTimeout = 120000;

    // set up the credentials for the basic authentication
    BasicAuthSecurityHandler basicAuthHandler = new BasicAuthSecurityHandler();
    basicAuthHandler.setUserName("foo");
    basicAuthHandler.setPassword("bar");

    // set up the HTTP connection
    final OslcRestClient oslcRestClient = new OslcRestClient(PROVIDERS,
            resourceURI, mediaType, readTimeout,
            basicAuthHandler);

    // retrieve the SysML block as POJO
    final SysMLBlock sysMLBlock = oslcRestClient
            .getOslcResource(SysMLBlock.class);
    System.out.println("SysML Block name: " + sysMLBlock.getName());
    System.out.println("SysML Block full ports:");
    for (Link sysMLFullPortLink : sysMLBlock.getFullPorts()) {
        System.out.println(sysMLFullPortLink.getValue());
    }
    System.out.println("SysML Block proxy ports:");
    for (Link sysMLProxyPortLink : sysMLBlock.getProxyPorts()) {
        System.out.println(sysMLProxyPortLink.getValue());
    }
}
```

```
SysML Block name: Wired Camera
SysML Block full ports:
http://localhost:8080/oslc4jmagicdraw/services/Wired_Camer
SysML Block proxy ports:
http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/proxyports/Blocks::Wired_Camera::ethernet
http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/proxyports/Blocks::Wired_Camera::power
http://localhost:8080/oslc4jmagicdraw/services/Wired_Camera_Example/proxyports/Blocks::Wired_Camera::video
```

OSLCWebClient4CreatingSysMLIBD.java: This client uses the OSLC services for adding SysML elements to existing MagicDraw projects (OSLC creation factories). The new OSLC resources are first described as POJOs using the OSLC4J library and then submitted to the OSLC creation services using HTTP POST requests. As an example, this Java client creates the elements that are displayed in the IBD below.



TestOSLCClientWithApacheWink.java: This client first retrieves the OSLC ServiceProviderCatalog resource as POJO and then accesses the service provider resources, the query services, and ultimately the OSLC resources.