**Instructions to install and run the OSLC MagicDraw SysML Adapter**

**(for v17.0.3, v17.0.4, v17.0.5, v18.0)**

By Axel Reichwein ([axel.reichwein@koneksys.com](mailto:axel.reichwein@koneksys.com)) January 25, 2016

# Installing OSLC4J

Follow the instructions in the document named “Instructions to install OSLC4J”. The document also contains instructions on how to use a proxy server with Maven and Eclipse.

# Download, import, and build svnkit-client and adapter.subversion projects

Clone **subversion-client** repository

1. Open the Git Repositories View (Window -> Show View -> type “Git Repositories” in the search field)
2. Click on the Clone Repository icon 
3. In the URI field, paste the following URL: <https://github.com/ld4mbse/subversion-client.git>
4. The Host and Repository fields will autofill. Enter your credentials in the Username and Password fields.
5. Click Next, only select the **master** branch
6. Click Next until Finish.

Clone **oslc-adapter-subversion** repository

1. Open the Git Repositories View (Window -> Show View -> type “Git Repositories” in the search field)
2. Click on the Clone Repository icon 
3. In the URI field, paste the following URL: <https://github.com/ld4mbse/oslc-adapter-subversion.git>
4. The Host and Repository fields will autofill. Enter your credentials in the Username and Password fields.
5. Click Next, only select the **master** branch
6. Click Next until Finish.

Import **svnkit-client project** into Eclipse workspace and build project

1. In the Git repositories view, right-click **svnkit-client** and select “Import Projects”. Click Next until Finish
2. The **svnkit-client** project is in the Eclipse workspace
3. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
4. Expand the **svnkit-client** project
5. Right click pom.xml -> Run As -> Maven clean
6. Right click pom.xml -> Run As -> Maven install

Import **edu.gatech.mbsec.adapter.subversion** into Eclipse workspace and build project

1. In the Git repositories view, right-click **edu.gatech.mbsec.adapter.subversion** and select “Import Projects”. Click Next until Finish
2. The **edu.gatech.mbsec.adapter.subversion** project is in the Eclipse workspace
3. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
4. Expand the **edu.gatech.mbsec.adapter.subversion** project
5. Right click pom.xml -> Run As -> Maven clean
6. Right click pom.xml -> Run As -> Maven install

# Downloading oslc-adapter-magicdraw-sysml repository

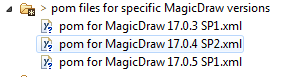
1. Open the Git Repositories View (Window -> Show View -> type “Git Repositories” in the search field). The view may be displayed at the right or in the lower middle section of your Eclipse IDE
2. Click on the Clone Repository icon 
3. In the URI field, paste the following URL: <https://github.com/ld4mbse/oslc-adapter-magicdraw-sysml.git>
4. Enter your Bitbucket credentials (username and password).
5. Click Next, Next again, and then Finish.

# Importing projects into the Eclipse workspace

1. In the Git repositories view, right-click edu.gatech.mbsec.adapter.magicdraw and select “Import Projects”. Click Next until Finish
2. 3 projects additional will be in the Eclipse workspace

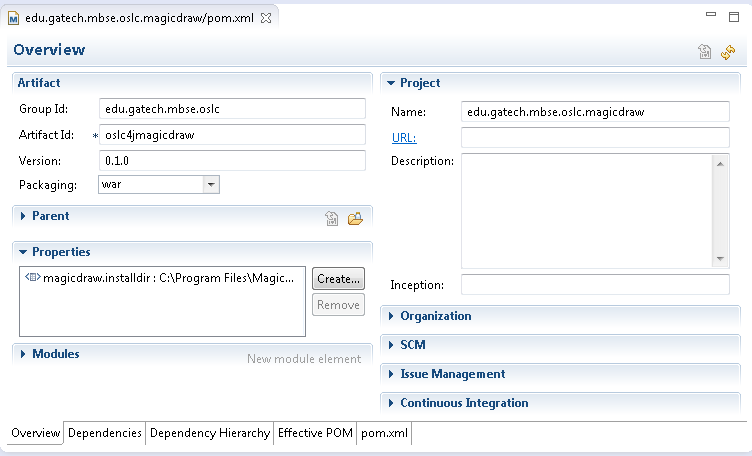
# Choosing the MagicDraw-specific pom file

1. By default, the pom.xml file is set up for MagicDraw 18.0.1 sp1. **If you are using MagicDraw 18.0, 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**,
   1. **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*”.
   2. Copy the file under the project root, so under the project named **edu.gatech.mbsec.adapter.magicdraw***.*
   3. *Then* **delete** the **pom.xml** file which was already under the project root,
   4. and **rename** the pom file you have just copied into the project root to **pom.xml**



# 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 **edu.gatech.mbsec.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.



1. 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.



# Building the edu.gatech.mbsec.adapter.magicdraw projects

1. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
2. Expand the **edu.gatech.mbsec.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 **edu.gatech.mbsec.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 **edu.gatech.mbsec.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.

1. If there is a red error mark next to any project, select the project. Right-click->Maven->Update Project… and click OK
2. 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.



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 1.8 is selected as Java version.

If a project still shows an error, then change its **JDK compliance** to 1.8. Select the project, right-click -> Properties. Select Java Compiler and select **1.8** in the drop down menu next to the JDK compliance setting as highlighted below.

# Manual configuration

TheOSLC MagicDraw adapter currently supports the retrieval of MagicDraw projects within

1. **a specific local directory**

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 *edu.gatech.mbsec.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/edu.gatech.mbsec.adapter.magicdraw/Magicdraw Models/**

Note: The file path can contain backslashes

**Warning**: Do not put quotes around the file path!

1. **or from a subversion repository**

Set the value of **syncWithSvnRepo** to **true**

Specify the **Subversion repository URL containing MagicDraw models** which will be considered by the OSLC MagicDraw SysML adapter in the config.properties file under *edu.gatech.mbsec.adapter.magicdraw/configuration*. As an example below, the **Subversion repository URL** is specified to be **https://myrepos.com/svn/magicdrawrepository/**

**Warning**: Do not put quotes around the file path!

Set the time period in seconds at which the adapter will poll the Subversion repository for updates. Example: **delayInSecondsBetweenDataRefresh** = none **or**

**delayInSecondsBetweenDataRefresh** = 90

Specify your Subversion credentials through the **svnUserName** and **svnPassword** fields

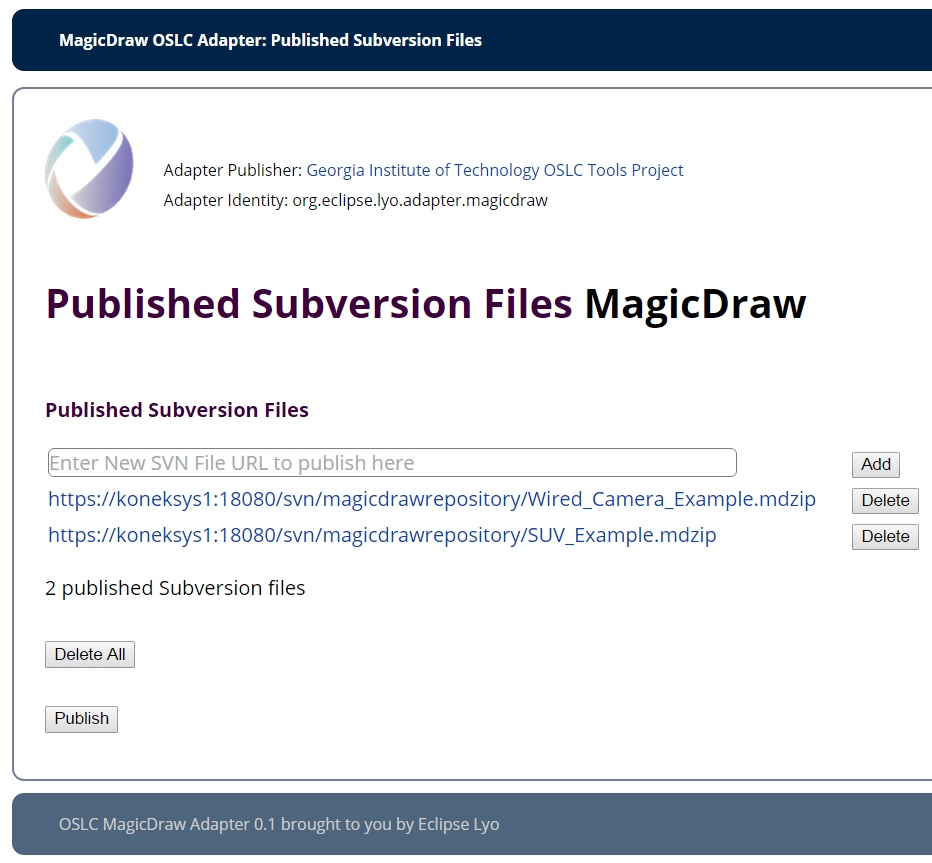
1. **or from individual Subversion-hosted files**

Set the value of **useIndividualSubversionFiles** to **true**

1. Specify the **Subversion file URLs representing MagicDraw models** which will be published by the adapter at startup in the subversionfiles.csv file under *edu.gatech.mbsec.adapter.magicdraw/configuration*. As an example displayed below, the **Subversion file URLs** are specified to be



1. **During adapter runtime, you can change the Subversion files** to be published through the web app **at** [**http://localhost:8080/oslc4jmagicdraw/services/svnfilepublisher**](http://localhost:8080/oslc4jmagicdraw/services/svnfilepublisher)



By clicking on **Publish**, the adapter will retrieve the latest version of the Subversion files and publish them

1. Specify your Subversion credentials through the **svnUserName** and **svnPassword** fields
2. Specify the **location of the folder containing MagicDraw models** where the Subversion files will be saved locally in config.properties file under *edu.gatech.mbsec.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/edu.gatech.mbsec.adapter.magicdraw/localworkingdirs**

The contents of this folder will be deleted when adapter starts in individualSubversionFile mode.

**Warning**: Do not choose as local Subversion file storage the same folder as the one containing all sample MagicDraw models, nor the one containing the local MagicDraw models models without Subversion info.

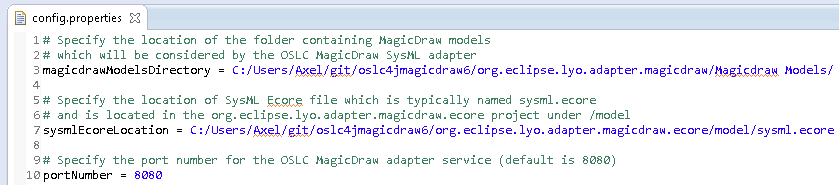
Several example MagicDraw projects are located in the ***MagicDraw Models*** folder in the **edu.gatech.mbsec.adapter.magicdraw** project. Following steps are also necessary to configure the adapter:

1. Specify the **location of SysML Ecore file** in the config.properties file under *edu.gatech.mbsec.adapter.magicdraw/configuration*. The location of the SysML ecore file named sysml.ecore is in the edu.gatech.mbsec.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/edu.gatech.mbsec.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!

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



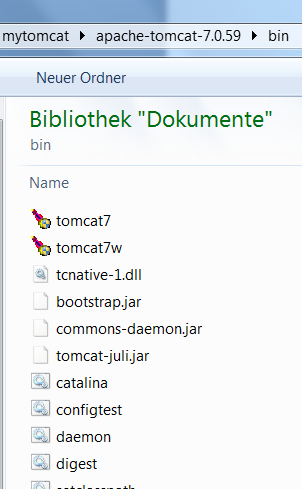
# Installing Apache Tomcat

1. Download Tomcat 7 by going to this page:

<https://tomcat.apache.org/download-70.cgi>

Note: Even though Tomcat 8 is available, it is preferable to use Tomcat 7 as the Tomcat Maven plugin as it only officially supports Tomcat7. It is still possible to use Tomcat7 running on Java8.

1. Download the zip distribution for your operating system. Note: do not use the windows installer as it doesn’t install all Tomcat scripts.
2. Unzip the Tomcat 7 distribution in a folder where your user account has read/write permission. Note: Windows disables direct file access to programs folder for normal users per default. Note: Installation of XAMPP can mess up previous installations of Tomcat and may need to be removed
3. Make sure that the /bin folder in your Tomcat installation directory contains the catalina.bat batch file as shown below

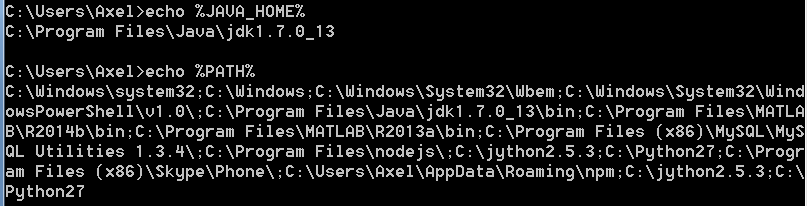


# Configuring Java for Tomcat

1. Make sure that you have JDK 8 installed on your machine. On Windows, verify your installed Java version by typing in the command prompt **java –version**



1. Test that environment variables JAVA\_HOME and PATH respectively point to JDK and JDK/bin. Verify this on Windows by typing in the command prompt **echo %JAVA\_HOME%** and **echo %PATH%**. If necessary set the envornoment variable in the command prompt using the **set** command (Example: **set** variable=string). After having set the environment variables, open a new ommand prompt to verify the values of the environment variables.



1. Make sure that JAVA\_HOME and JRE\_HOME both point to the same Java version, for example Java 8. Adapters are now currently being build with Java 8 compilers. So Tomcat should also run with Java 8.
2. Make sure that CATALINA\_HOME points to the correct installation directory of your Tomcat distribution.

# Enabling PUT on Apache Tomcat

Tomcat by default is not enabled for HTTP PUT command. But, it can easily be configured to support it.

1. In your Apache Tomcat 7 installation directory, open **/conf/web.xml**
2. Add the ***readonly* init param** to the web.xml file as shown below and save the file

**<init-param>**

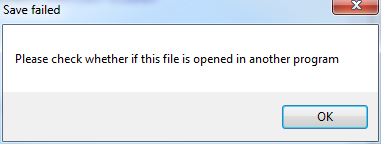
**<param-name>readonly</param-name>**

**<param-value>false</param-value>**

**</init-param>**

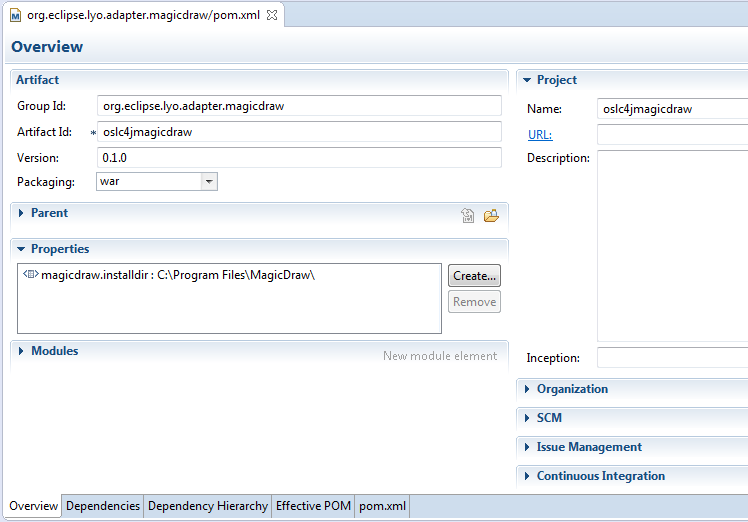


**Note**: If you get the warning shown below while trying to save the file, then copy the web.xml file into another location, modify it, and then replace the original web.xml file by the modified web.xml file.



# Setting the Apache Tomcat server port

1. By default, the OSLC MagicDraw adapter service will run on port 8080. Change the port of the oslc4jmagicdraw adapter service only if you need to avoid a conflict with another service already running on port 8080. Skip the next steps if you do not need to change the port.
2. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer)
3. Expand the **edu.gatech.mbsec.adapter.magicdraw** project
4. Select and open the maven **pom.xml** file through double-click
5. 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. Click on the **pom.xml tab** of the pom.xml file as highlighted below.

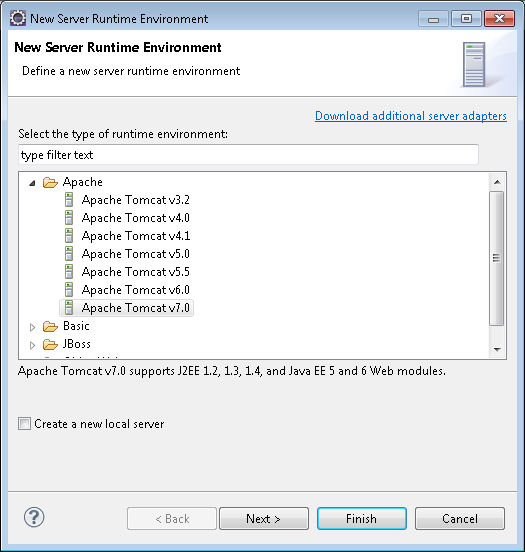


1. In the pom.xml tab of the pom.xml file, specify the **port of the OSLC MagicDraw adapter service** in theMaventomcat plugin configuration found at the **bottom of the pom.xml** tab of the pom.xml file as highlighted below. Enter the port number in the configuration section as shown below.



# Adding Server Runtime Environment in Eclipse

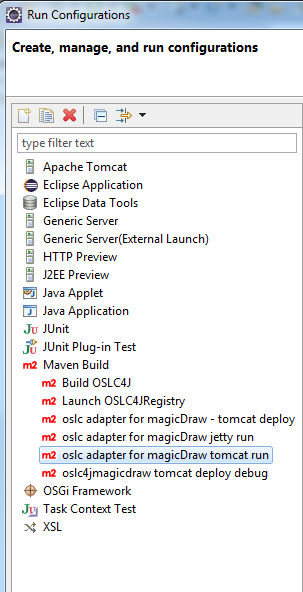
1. In Eclipse. Open **Window -> Preferences -> Server -> Runtime Environments** to create a Tomcat installed runtime.
2. Click on **Add...** to open the New Server Runtime dialog,
3. From the drop down menu, select Tomcat 7.0 as shown below. Click Next.

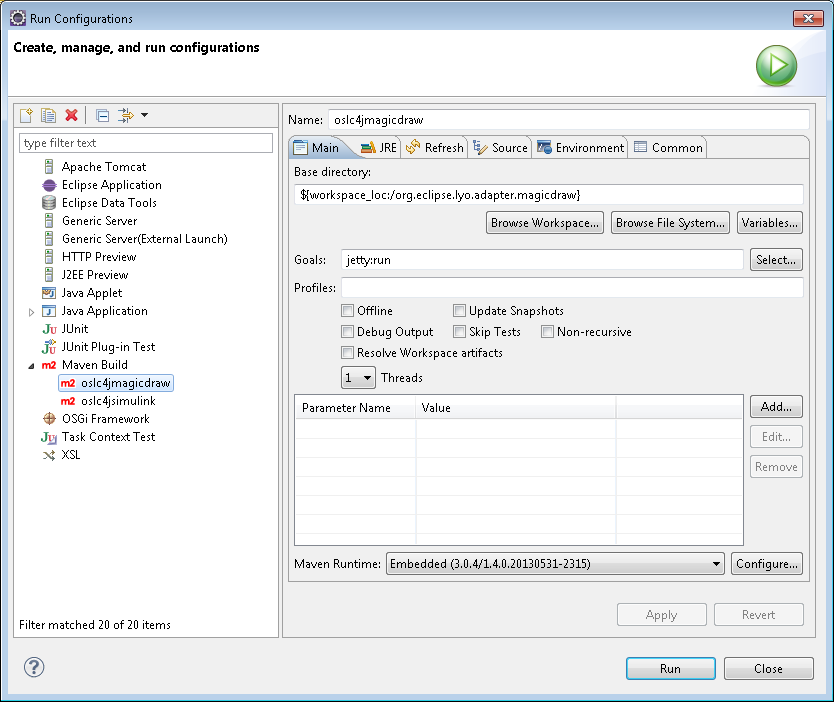


1. Enter the **Tomcat 7.0 installation directory** (not the Apache installation directory!) as highlighted below.
2. Click on Finish.

# Creating a launch configuration for edu.gatech.mbsec.adapter.magicdraw

1. Open the launch configuration window (Run -> Run Configurations…)
2. Select the **Maven Build** category and click on **oslc adapter for magicDraw tomcat run**





1. **Only If you want to assign more memory to the OSLC MagicDraw adapter**, place your memory settings In the JRE tab, under VM arguments, such as:

**-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):

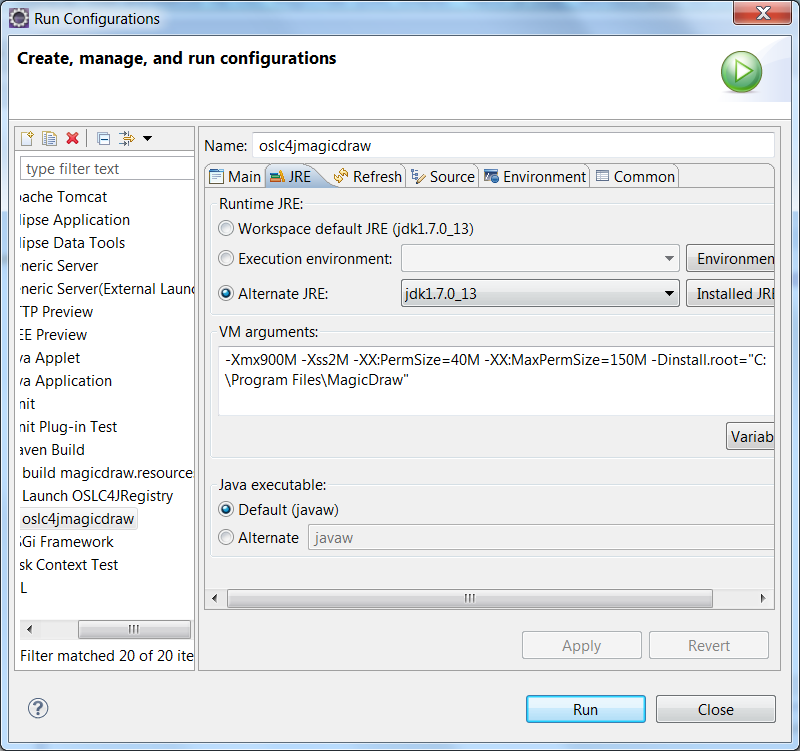
**-Xmx3674M -XX:PermSize=60M -XX:MaxPermSize=1350M**

**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**



# 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>

# Launching edu.gatech.mbsec.adapter.magicdraw (OSLC MagicDraw Adapter)

Select the oslc4jmagicdraw launch configuration (Run -> Run Configurations… and select in the Maven build category the launch configuration named **oslc adapter for magicDraw tomcat run**) and click Run.

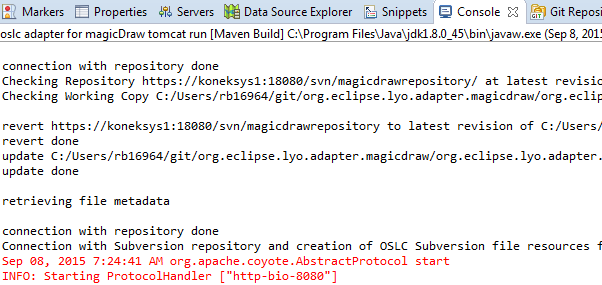
**Note**: The Tomcat server will launch. This will take a few minutes because it has to load many jars (especially all the MagicDraw jars).

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

The OSLC4 MagicDraw adapter is running when you can see the line

**INFO: Starting ProtocolHandler ["http-bio-8080"]**

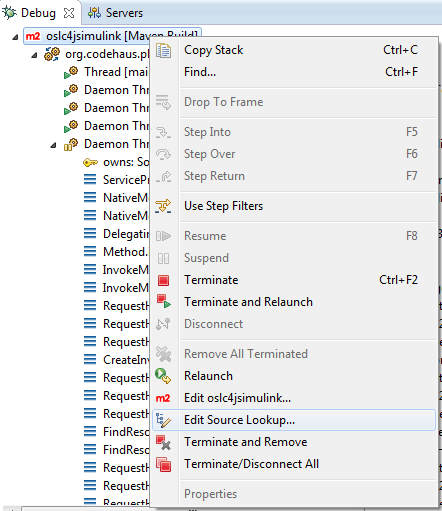
In the console window as shown below.

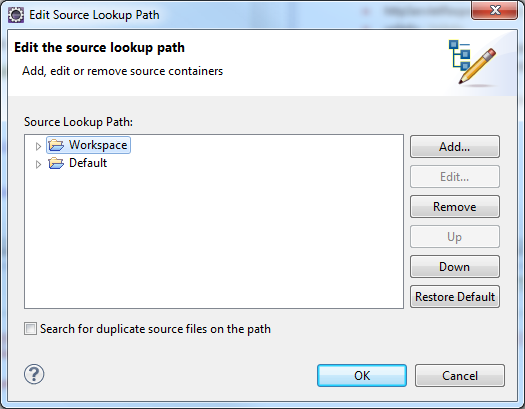


**Warning**: If the Tomcat server fails to launch due to a **java.net.BindException**, a different port for the OSLC MagicDraw adapter needs to be used since there is a conflict with another service using the same port. By default, Jetty uses port 8080. A **java.net.BindException** means that a different 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.

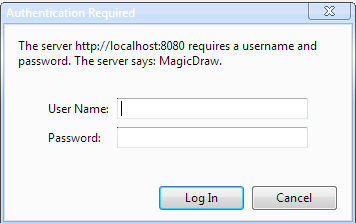




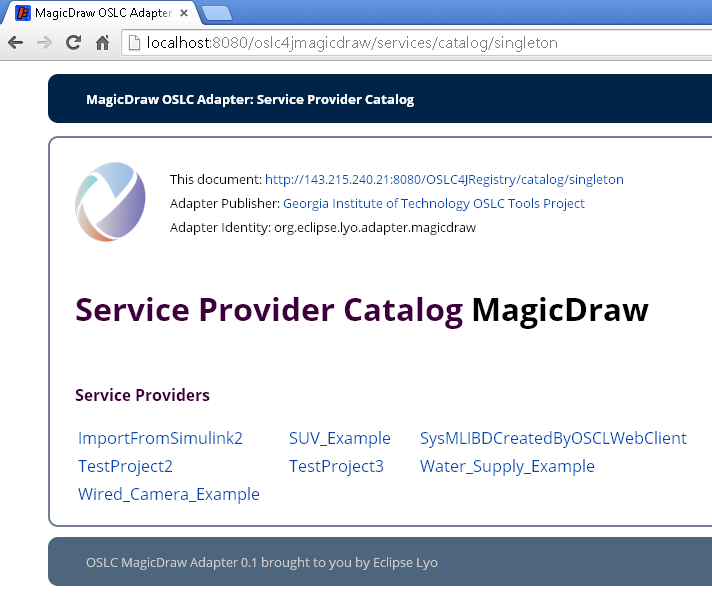
# Testing the OSLC MagicDraw Adapter

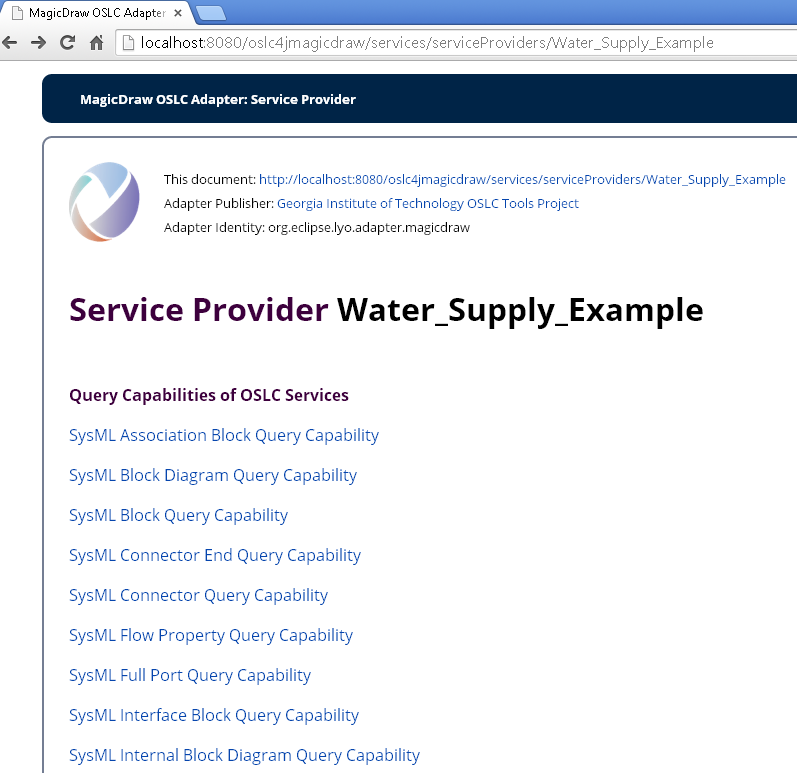
**Testing the retrieval of OSLC resources in HTML**

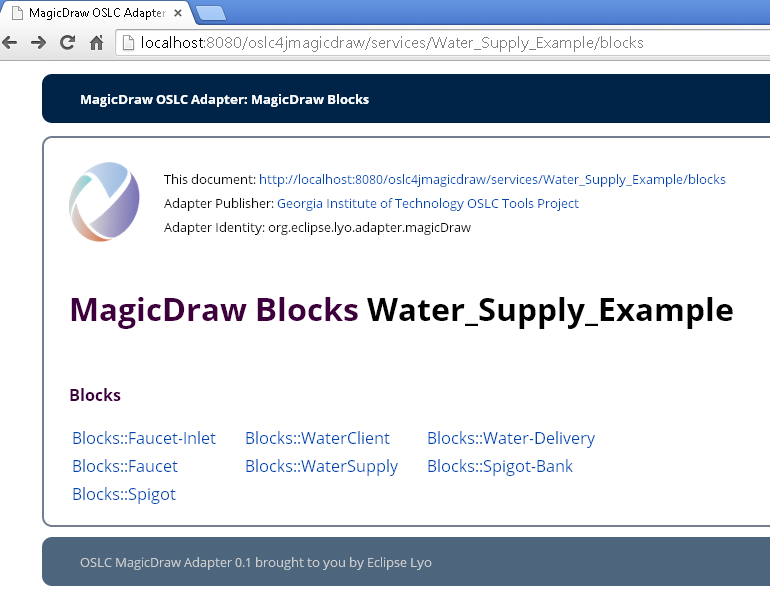
1. Launch Google Chrome
2. In the URL field, type for test purposes: <http://localhost:8080/oslc4jmagicdraw/services/catalog/singleton>. This will send a HTTP GET request to retrieve the HTML representation of the MagicDraw Service Provider Catalog
3. You may need to authenticate yourself. If you do, 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.

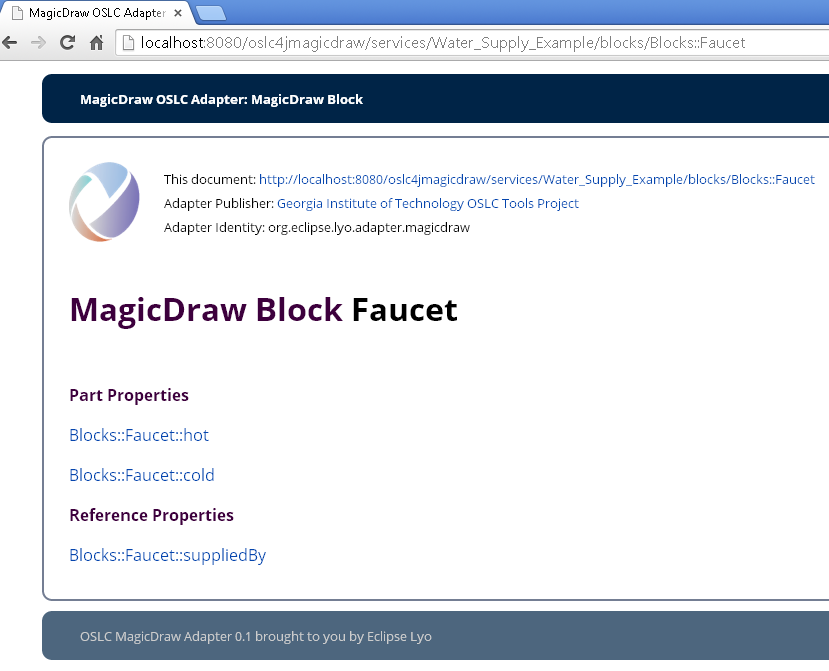


1. You will then see an HTML page showing you the list of Service Providers. You can browse from the Service Providers (e.g. for TestProject2) to the Services and ultimately to the SysML resources.





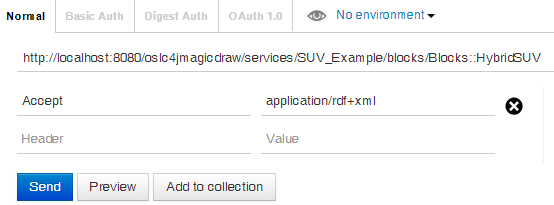




**Testing the retrieval of OSLC resources in RDF**

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> 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.

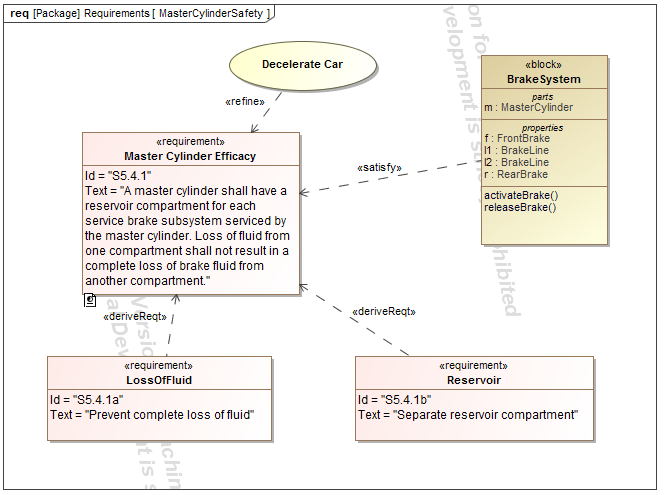




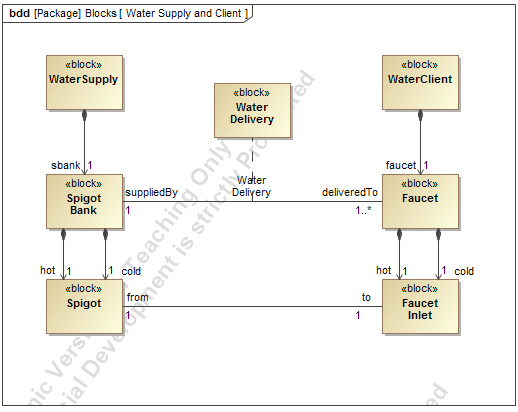
# Testing the OSLC MagicDraw Adapter through example MagicDraw models

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

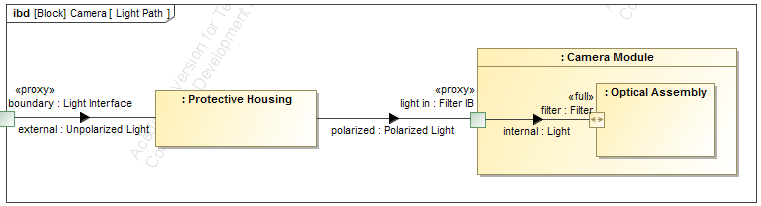
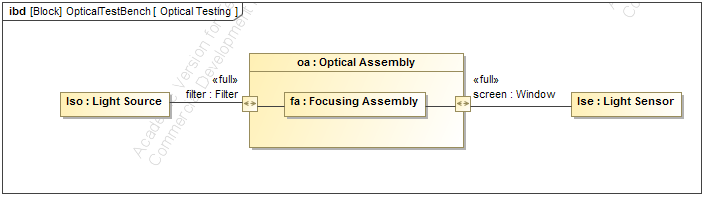
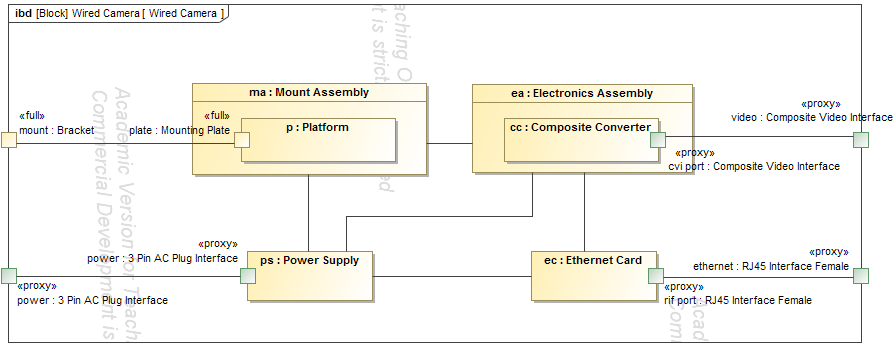
**SUV Example** contains requirements and relationships between requirements and different SysML elements.



**Water Supply Example** contains blocks, associations, and associationblocks.



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

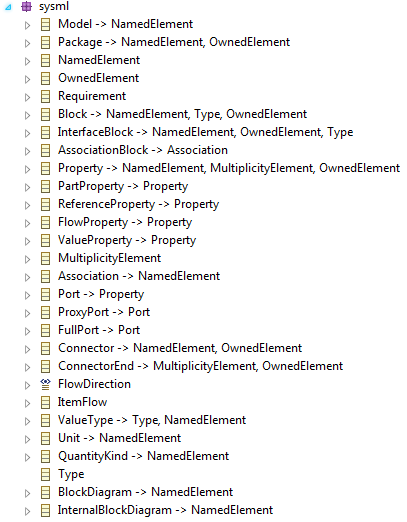


# Getting an overview of SysML concepts supported by the OSLC MagicDraw adapter

All the concepts supported by the OSLC MagicDraw adapter have been defined in an Ecore metamodel named **sysml.ecore**, located in the project named *edu.gatech.mbsec.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.

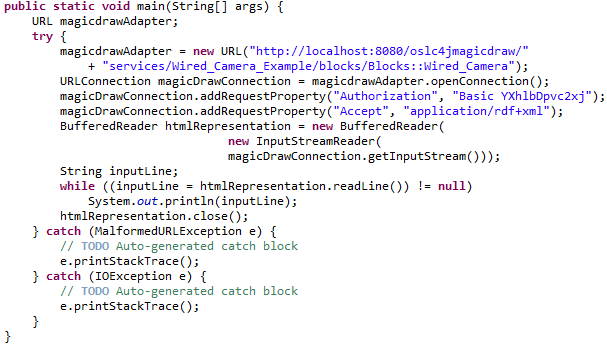
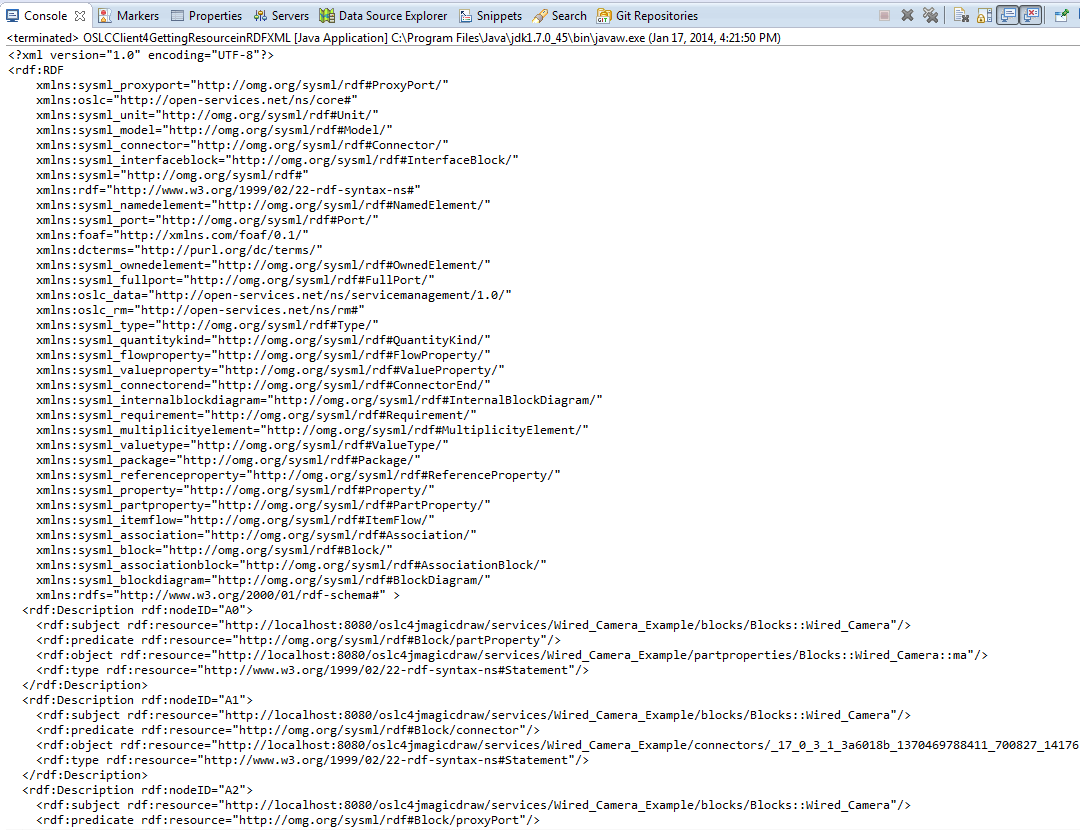
Not all concepts are supported equally. As an example, the adapter has an OSLC service to create blocks, but there is no service for creating interface block diagrams. In addition, abstract SysML concepts such as NamedElement, which are reused across several SysML concepts, are not supported by the OSLC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



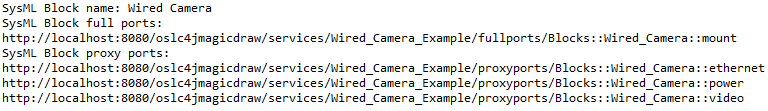
# Accessing SysML elements as OSLC resources using Java clients

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 *edu.gatech.mbsec.adapter.magicdraw.clients* package.

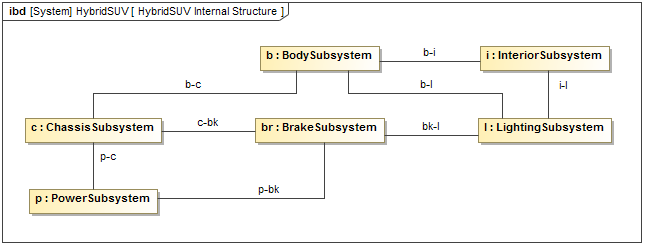
**HTTPClient4GettingResourceRepresentation**.java: This Java client retrieves an OSLC resource and prints into the console the representation of the OSLC resource.



**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.



**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.