Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 5.2

Installation and Configuration Guide
with IBM WAS Liberty Platform

Copyright and Legal Notice

Table of Contents

Revision History

Revision	Release Date	Notes
1.0	March 2016	Initial release of this document.
Last modified: March 2016		

1.   Dialogic JSR 309 Connector Requirements

The following requirements are needed before installing the Dialogic JSR 309 Connector:

- A functional IBM WAS Liberty Profile Application Server platform for development and testing.

  The Dialogic JSR 309 Connector has been tested with: Liberty Beta version found at:

  https://developer.ibm.com/wasdev/downloads/liberty-profile-beta

- A functional PowerMedia XMS Release 3.1 system.

- SIP phones and/or soft clients.

2.   Contents of the Distribution

This section lists and describes the files in the Dialogic JSR 309 Connector distribution.

The Dialogic JSR 309 Connector distribution consists of a single TAR file:

dialogic309-M.m.BBBB-liberty.tar

Where:

- *M* stands for a major version number.
- *m* stands for a minor version number.
- *BBBB* stands for a build number.

This package contains the following structure.

Content	Description
DIR: */DlgcJSR309/application/* CONTENTS: *Dialogic_sample_demo.war* *Dialogic.mp4* *Project/*	Directory that contains Dialogic JSR 309 Verification Application *dlgc_sample_demo.war* ready to be deployed and the *Dialogic.mp4* media file used by the Verification Application (which will be part of upcoming XMS installs). Directory also contains the project directory, which has all of the necessary items to build *dlgc_sample_demo.war*.
DIR: */DlgcJSR309/Dlgc309Connector/* */DlgcJSR309/3rdPartyLibs/*	Directory that contains the Dlgc309Connector, which has all of the Dialogic connector files, and the 3rdPartyLibs directory, which has all necessary third-party JAR files.
DIR: */DlgcJSR309/properties/* CONTENTS: *Dialogic-Samples.war* *dar.properties* *log4j2.xml*	Directory that contains Verification Application properties files used to set up its configuration and the configuration parameters for Dialogic JSR 309 Connector. Directory that contains the *dar.properties* file, which is used for SIP routing. Directory also contains the *Log4j2.xml* log configuration file used for Dialogic JSR 309 Connector and Verification Application logging.

3. Installation and Configuration

This section describes how to install and use the Dialogic JSR 309 Connector.

The following steps are necessary for Dialogic JSR 309 Connector and demo application installation for correct operation:

- Preparing the J2EE Converged Application Server
- Installing the Dialogic JSR 309 Connector
- Configuring the IBM Liberty for the Dialogic JSR 309 Connector
- Installing the Dialogic JSR 309 Connector User Feature
- Installing and Configuring the Dialogic JSR 309 Connector Verification Application
- Configuring the PowerMedia XMS Media File
- Running the Dialogic JSR 309 Verification Application

For system requirements and supported platforms, see Dialogic JSR 309 Connector Requirements.

Preparing the J2EE Converged Application Server

The Dialogic JSR 309 Connector has been deployed and tested on specific versions of IBM WAS Liberty Application Servers. For quick instructions on how to install and configure the desired Application Server (AS) for Dialogic JSR 309 Connector usage, refer to Appendix A: Dialogic JSR 309 Connector Environment Setup.

Installing the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is configured as a User Feature in IBM Liberty Application Server.

The Dialogic JSR 309 Connector demo application provided with the distribution is used for simple verification of correct installation and configuration of Dialogic JSR 309 and required XMS. Its source code is also provided to illustrate functionality used. Further details of this verification application can be found in Test Servlets section in this document.

The distribution package needs to be extracted on a local system because various components (files) will be needed to correctly complete each step. Refer to Contents of the Distribution, which describes the contents in detail.

Place the Dialogic connector package TAR file on IBM Liberty application platform where it can be easily accessible. Then, run the following command:

```
tar –xvf dialogic309-M.m.BBBB-liberty.tar
```

This will create a directory, *DialogicSampleDemo,* as described in Contents of the Distribution.

Note: These directory is referenced throughout this document for content required by Dialogic JSR 309 Connector.

Configuring the IBM Liberty for the Dialogic JSR 309 Connector

This section contains the IBM Liberty configuration prerequisites. To begin, edit *server.xml* found at:

```
<Home Dir>/wls/usr/server/<serverName>/server.xml
```

Add the following lines marked in **RED**:

```
<?xml version="1.0" encoding="UTF-8"?>

<server description="new server">


    <!-- Enable features -->

    <featureManager>

        <feature>sipServlet-1.1</feature>

        <feature>mediaServerControl-1.0</feature>

        <feature>websocket-1.1</feature>

        <feature>usr:com.vendor.dialogic.javax.media.mscontrol.LIBERTY.snapshot</feature>

    </featureManager>


        <!-- This template enables security. To get the full use of all the capabilities, a
keystore and user registry are required. -->


        <!-- For the keystore, default keys are generated and stored in a keystore. To provide
the keystore password, generate an

            encoded password using bin/securityUtility encode and add it below in the password
attribute of the keyStore element.

            Then uncomment the keyStore element. -->

        <!--

        <keyStore password=""/>

        -->


        <!--For a user registry configuration, configure your user registry. For example,
configure a basic user registry using the

            basicRegistry element. Specify your own user name below in the name attribute of the
user element. For the password,

            generate an encoded password using bin/securityUtility encode and add it in the
password attribute of the user element.

            Then uncomment the user element. -->

        <basicRegistry id="basic" realm="BasicRealm">

                <!-- <user name="yourUserName" password="" />  -->

        </basicRegistry>


    <!-- To access this server from a remote client add a host attribute to the following
element, e.g. host="*" -->

    <httpEndpoint id="defaultHttpEndpoint"

                httpPort="9080"

                httpsPort="9443" />


    <!-- Automatically expand WAR files and EAR files -->
```

```
        <applicationManager autoExpand="true"/>


<sipApplicationRouter
sipDarConfiguration="/home/ibm/wlp/usr/servers/defaultServer/resources/Dialogic/dar.properties"
/>

<sipStack pathMtu="40000" />


</server>
```

The modifications are required for the Dialogic JSR 309 Connector and for any application that will use the connector.

- featureManager - These features are needed for an application when used with the Dialogic JSR 309 Connector user feature. Optionally, the websocket-1.1 feature needs to be specified if the application will use websockets and not used by the Dialogic JSR 309 Connector itself. Note that having other features might interfere with other required features. Consult IBM Liberty support if required features prevent from proper startup and operation.

- sipApplicationRouter - This is used to define how SIP requests should be routed. Used by Dialogic verification application.

- sipStack - pathMtu size might need to be increased as the use of SDP for WebRTC clients might exceed default size. Therefore it is recommended for the mtu size to be adjusted.

Installing the Dialogic JSR 309 Connector User Feature

To install the Dialogic JSR 309 Connector user feature on the IBM Liberty application server, the following steps need to be taken:

1. Place Dialogic JSR 309 Connector user feature file (.esa file) on the Liberty system (for example, the */tmp* directory).

2. Install the user feature running the following command from the *wlp/bin* directory:
   ```
   ./featureManager install
   /tmp/com.vendor.dialogic.javax.media.mscontrol.LIBERTY.snapshot_5.0.1.esa
   ```

Successful feature installation will produce the following output:

```
Step 1 of 5: Starting installation ...

Step 2 of 5: Checking features ...

Step 3 of 5: Installing com.vendor.dialogic.javax.media.mscontrol.LIBERTY.snapshot ...

Step 4 of 5: Cleaning up temporary files ...

Step 5 of 5: Installation completed


All features were successfully installed.


Start product validation...

Product validation completed successfully.
```

Installing and Configuring the Dialogic JSR 309 Connector Verification Application

An application can take advantage of the Dialogic JSR 309 Connector and use its resources for media related functionality. The Dialogic JSR 309 Connector package provides a verification application that uses the Dialogic JSR 309 Connector. This demo is used to

illustrate correct configuration of the platform, the Dialogic JSR 309 Connector, and Dialogic PowerMedia XMS. Install and configure the Dialogic JSR 309 Connector verification demo application as follows:

1. Configure the Environment Variables
2. Configure the Dialogic JSR 309 Connector Verification Application Properties File
3. Deploy the Dialogic JSR 309 Connector Verification Application

Configure the Environment Variables

In order for Dialogic JSR 309 Verification demo to operate correctly the following changes need to be made inot .bashrc file as marked in **RED**

```
.bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi


export SERVER_HOME="/home/ibm/wlp/usr/servers/defaultServer/resources"
export SAMPLE_PROPERTY_FILE=${SERVER_HOME}/Dialogic/dlgc_sample_demo.properties


# User specific aliases and functions
```

Configure the Dialogic JSR 309 Connector Verification Application Properties File

The verification demo properties file contains various settings used by the application. Since the application is responsible for configuring the Dialogic JSR 309 driver instance, there are various settings that need to be correctly defined.

From the distribution package under *DlgcJSR309/properties*, copy the *dlgc_sample_demo.properties* file to the following directory, which will need to be created:

```
<Root Dir>/wlp/usr/servers/<serverName>/resources/Dialogic
```

Edit the *dlgc_sample_demo.properties* file and assure that following items marked in **RED** are configured approprietly to the installed environment:

```
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5060
connector.sip.transport=udp
mediaserver.sessionTimer.switch=off
mediaserver.sessionTimer.maxTimeout=120
mediaserver.sessionTimer.minTimeout=100
mediaserver.sip.address=xxx.xxx.xxx.xxx
mediaserver.sip.port=5060
play.prompt=file:////en_US/verification/Dialogic.mp4
dlgc.jsr309.driver.name=com.dialogic.dlg309
```

Note the following:

- connector.sip.address - The Dialogic JSR 309 Connector automatically detects an IP available from the platform. However, if the platform has more then one network interface configured, it is up to the application to choose the right interface. This parameter provides the ability to overwrite the automatically chosen IP address.

- mediaserver.ip.address - Specify the IP address of the PowerMedia XMS server to be used by the Dialogic JSR 309 Connector.

Deploy the Dialogic JSR 309 Connector Verification Application

The Dialogic JSR 309 Connector Verification Application needs to be deployed in the IBM Libyerty Application Server. To do so, copy the application .war file into following directory:

```
<Root Dir>/wlp/usr/servers/<serverName>/dropins
```

Configuring the PowerMedia XMS Media File

The Dialogic JSR 309 Verification Application has been developed to use the *Dialogic.mp4* media file. This media file will become part of Dialogic PowerMedia XMS distribution; however, as of PowerMedia XMS release 3.0 Service Update 1, the media file is not part of distribution and must be installed manually.

1. To install *Dialogic.mp4* manually, log in to PowerMedia XMS WebGUI, and then click **Media**.

2. Click the **Media Management** tab.
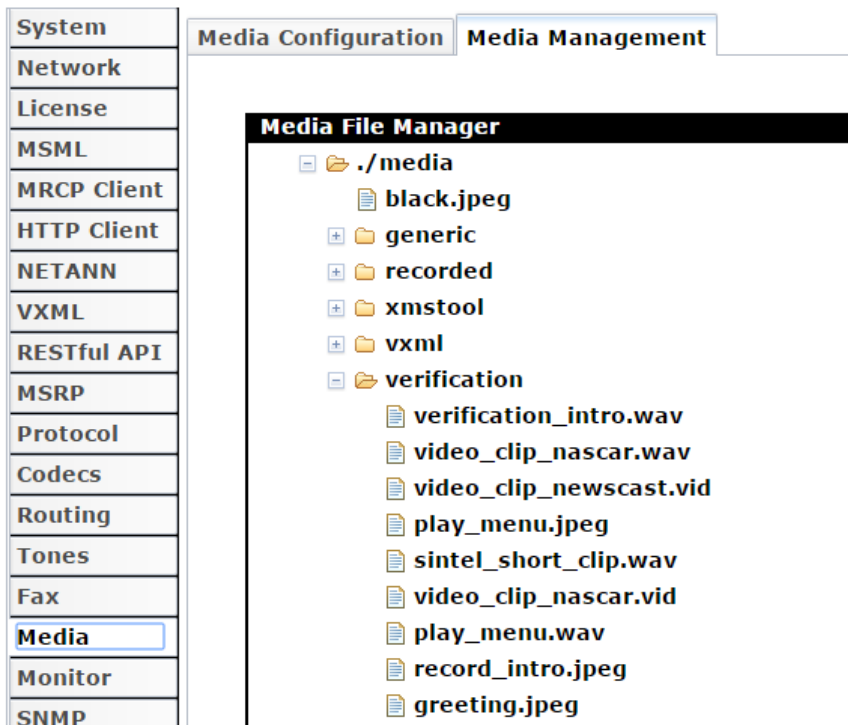
3. In the **Media File Manager** section, Right-click the **verification** folder and click **Upload Media File**.

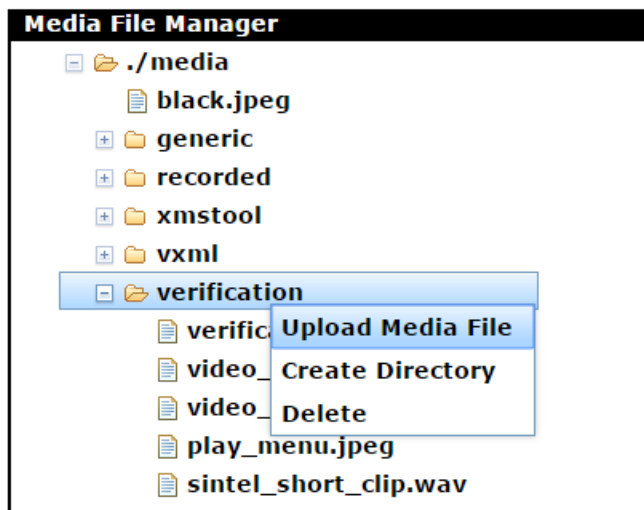4. Click **Browse**, select *Dialogic.mp4* in the Dialogic JSR 309 Connector distribution folder, and click **Upload**.
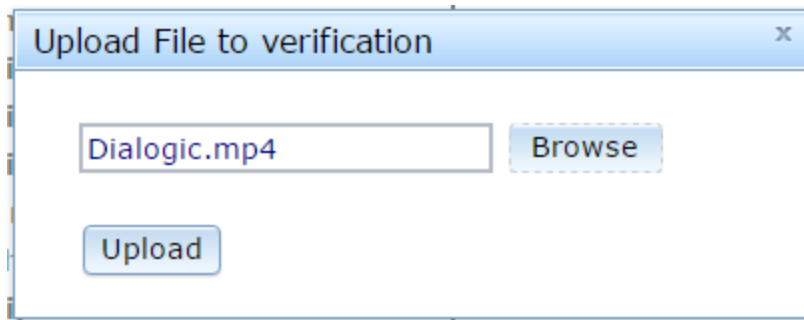
Once uploaded, the *Dialogic.mp4* media file is stored in the appropriate location for the Dialogic JSR 309 Verification Application to use it as per its configuration.
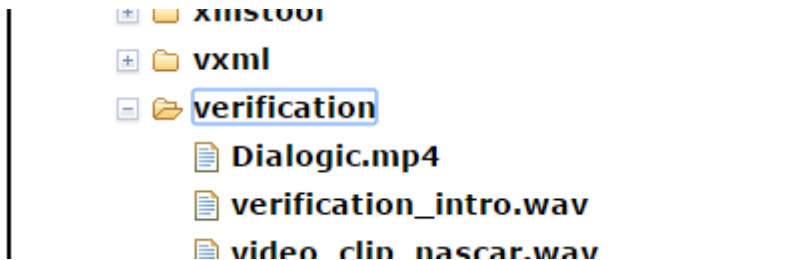
Running the Dialogic JSR 309 Verification Application

The Dialogic JSR 309 Verification Demo (player) has been written to support both SIP and WebRTC (Chrome and Firefox only at this time) clients.

SIP Client:

1. Have a SIP client configured for supported audio codec.

2. Place a call into the Application Server with the following URI:

   ```
   player@<as_ip_address>
   ```

   With successful configuration, the sample verification .mp4 should be heard and/or seen.

WebRTC Client:

Note: With the newest versions of browsers, WebRTC media functionality and WebSocket support are only allowed via a secure web connection (HTTPS). The J2EE platform will need to be configured for HTTPS in order for a WebRTC Client to work. Follow the online documentation of the platform for HTTPS configuration instructions.

1. Open a Chrome or Firefox web browser.

2. Navigate to the following URL:

   ```
   <AS_IP_Address>:9080/Dialogic-Samples/DemoGUI/index.html
   ```

3. Click **Play**.

4. Dialogic JSR 309 Verification Application

About

The Dialogic JSR 309 Verification Application is provided with each platform specific package for two reasons:

1. The application (WAR file), which uses the Dialogic JSR 309 Connector, is provided as a tool to verify the Application Server platform and Dialogic PowerMedia XMS operation.

2. The application project source has all the necessary components required to create a platform-specific application using the Dialogic JSR 309 Connector. This can quickly help clarify various steps that are required in the J2EE application using the Dialogic JSR 309 Connector. It includes the following:

   a. Provides steps on how to create an application (WAR file) to run in a specific J2EE AS platform

   b. Illustrates application initialization steps

   c. Illustrates application initialization steps necessary for use with the Dialogic JSR 309 Connector

   d. Illustrates the steps the application needs to take in order to work with SIP and/or web based multimedia (WebRTC) clients (provided that the chosen platform provides support for server side WebSockets).

The Details

This section details the different areas of the Verification Application for better understanding of the basic, necessary steps for any application.

- Application WAR File Content
- Application Initialization Steps
- Application Steps to Initialize the Dialogic JSR 309 Connector

Application WAR File Content

Minimum content of the application is illustrated in Dialogic JSR 309 Verification Application war file. The war package contains several necessary items. Please refer to build.xml to get yourself familiar with how the war file is generated.

The *dlgc_sample_demo.war* file consists of three directories:

- The */META-INF* directory contains a *MANIFEST.MF,* which is a standard way of providing information about the package that contains it.
- The */WEB-INF* directory contains the following:
  - The "classes" directory, which contains JAVA .class files.
  - The "lib" directory, which contains all JAR files required by the deployment application WAR file.
  - The deployment structure .xml file is used to exclude some automatic dependencies.
  - The *sip.xml* file, which defines SIP servlet-mapping for the deployment application WAR.

14

- The */DemoGUI* directory contains content for the application that supports WebRTC. It includes necessary .html and .js files for the verification demo.

Application Initialization Steps

Below is an illustration of basic application structure used in this platform. For further details please reference a source of Dialogic JSR 309 Verification Application.

```
package play;


@ServerEndpoint("/base/{id}")


@SipListener
public class AsyncPlayer extends SipServlet implements Serializable, SipServletListener
{

        @Override
        public void init(ServletConfig cfg) throws ServletException

        {

        }


        @Override
        public void servletInitialized(SipServletContextEvent evt)

        {

        }
```

Note the following details:

- ServerEndpoint defines this class to be used to serve WebSocket requests.

- When the platform starts the application, it will invoke an init( ) function. This function should contain application specific initialization procedures. This is where the Verification Application reads the application properties file and stores its content in local storage to be used later when initializing the JSR 309 interface.

- Once the platform's SIP container is started, it will call the application's servletInitialized( ) method to inform it that the SIP stack is now ready for application usage. At this stage, the application can start to initialize the Dialogic JSR 309 Connector.

Application Steps to Initialize the Dialogic JSR 309 Connector

Once the application's servletInitialized( ) method is invoked, the SIP container has been initialized and the application can now take steps to initialize the JSR 309 interface. After validating the request in the Verification Application servletInitialized( ) method, the application will issue initDriver( ) method.

The application obtains the Dialogic JSR 309 Connector configuration from the application properties file::

```
protected boolean initDriver()

{

  demoPropertyObj = new ConfigProperty(this.getClass());

…

}
```

```
public class ConfigProperty  {


      protected Properties demoProps;
      final String SAMPLE_PROP_NAME="SAMPLE_PROPERTY_FILE" ;
…
}
```

The connector driver is able to discover some of the parameters that it needs but not all. The parameters required by the driver to work correctly are as follows:

- connector.sip.address – Platform SIP IP address used by the SIP container. The connector provides the ability to change the address in case the platform has multiple IP interfaces and the default IP address picked by connector needs to be changed.

- connector.sip.port - Platform SIP port address used by SIP container. The connector provides ability to change the address in case the platform has multiple IP interfaces and the proper one is defined for different port number.

- connector.sip.transport – Platform SIP transport. Supported values are "udp" or "tcp". Default: "udp".

- mediaserver.sip.ipaddress – Dialogic XMS Media Server SIP IP address to be used by the Dialogic JSR 309 Connector.

- mediaserver.sip.port - Dialogic XMS Media Server SIP port to be used by the Dialogic JSR 309 Connector.

Optionally, the Dialogic JSR 309 Connector supports turning on SIP Session Timers between the JSR 309 driver and the Dialogic PowerMedia XMS. In this version of the JSR 309 Connector, the SIP Session Timers are turned on by default. The application can modify the parameters for the SIP Session Timers when configuring factory properties:

- mediaserver.sessionTimer.maxTimeout – defines SIP Session timeout in seconds. Default: 1800 (seconds).

- mediaserver.sessionTimer.switch - Turns the SIP Session Timer function on or off. Allowed values are: "on" or "off". Default: "on".

The Verification Application configures the Dialogic JSR 309 Connector properties with appropriate configuration parametrs and its values:

```
…
  Properties factoryProperties = new Properties();
  for ( PropertyInfo prop: connectorProperty ) {
    log.debug("initDriver() - ===================");
    log.debug("initDriver() - Name: " + prop.name);
    log.debug("initDriver() - Description: " + prop.description);
    log.debug("initDriver() - Required: " + new Boolean(prop.required).toString() );
    log.debug("initDriver() - Value: " + prop.defaultValue);
    if ( prop.name.compareToIgnoreCase("connector.sip.address") == 0 )
    {
      if (prop.defaultValue.compareToIgnoreCase(new_connector_sip_address.toString()) != 0)
      {
        log.debug("initDriver() - New Value: " + new_connector_sip_address);
        prop.defaultValue = new_connector_sip_address;
      }
```

```
……

   factoryProperties.setProperty(prop.name, prop.defaultValue);

 }

…..
```

The application creates a new properties factory in which it will store all required parameters for the Dialogic JSR 309 Connector to start properly. It reads the locally stored application properties file configuration of each required parameter and compares it to the value automatically picked up by the Dialogic JSR 309 Connector. It then takes the newest value for each of the required parameters and stores it in new properties factory.

The Dialogic JSR 309 Connector factory is created with the new set of parameters.

```
....

 mscFactory = dlgcDriver.getFactory(factoryProperties);

….
```

The Dialogic JSR 309 Connector factory (mscFactory) is now created with a new set of required parameters. Now, the Dialogic JSR 309 Connector interface can be used.

5. Troubleshooting

This section provides basic troubleshooting techniques for the Dialogic JSR 309 Connector.

Logging

The Dialogic JSR 309 Connector and its sample applications use the Apache Log4j2 logging facility. Connector makes use of *log4j2.xml* file for its logging configuration. This *log4j2.xml* configuration file needs to be part of applications WAR file in the *WEB-INF\classes* directory. The *Dialogic.log* connector log output file can be found in the <Root Dir>/wlp/usr/servers/<ServerName>/logs directory.

Refer to the following to see *Log4j2.xml* in detail.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration monitorInterval="10" status="ERROR">
  <Appenders>
    <File name="dialogic" fileName="logs/Dialogic.log" append="false">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"/>
    </File>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"/>
    </Console>
  </Appenders>


  <Loggers>
    <Logger name="com.vendor.dialogic" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="play" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="base" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
  </Loggers>
</Configuration>
```

For details of the *Log4j2.xml* configuration, refer to the following information:

- monitorInterval – Parameter defines how often log4j2 facility will automatically detect changes to the configuration file and reconfigure itself. The default is 10 seconds.
    - Appenders:
        - Provided *log4j2.xml* file defines two streams (Appenders) that it will send logging to: a file (*Dialogic.log*) and a system console. Each individual logger has a choice of which appender to use.
    - Loggers:
        - Provided *log4j2.xml* file Loggers section provides a logger configuration for various Java source packages:
            - com.vendor.dialogic is a Dialogic JSR 309 Connector.
            - play & base is a Dialogic JSR 309 Connector Verification Application.

            Note: Each logger can be set individually to the appropriate level of logging and each logger can be individually configured to log to file, STDOUT, or both.

Note that default logging level is set to *ERROR,* which will cause the *Dialogic.log* file to be empty unless there are errors.

Refer to the Apache Log4j 2 documentation at http://logging.apache.org/log4j/2.x for details.

Additional platform component logging, configuration, and modifications can be accomplished via appropriate Application Server Administration page. Refer to the platform specific documentation for details.

SIP Errors

If the PowerMedia XMS returns "503 Service Unavailable", make sure the network is correctly set up by performing the following actions:

1. Verify the available PowerMedia XMS licenses.

2. Check the */etc/hosts* file configuration.

3. Make sure application properties file (i.e., *dlgc_sample_demo.properties*) is referencing the appropriate PowerMedia XMS and Application Server IP address and ports.

6. Building and Debugging Sample Demos in Eclipse IDE

The Dialogic JSR 309 Connector distribution package comes with all necessary configuration files and content needed for anyone to build the Verification Application on their own. This section provides the steps to create, compile, build, and debug provided demo application using Eclipse IDE.

Prerequisites

The following components must be installed:

- Latest JDK version supported by desired J2EE platform

- Eclipse KDE

- In order to build provided demo applications, obtain platform dependent libraries that are NOT provided as part of Dialogic JSR 309 Connector distribution package:

    o *mscontrol.jar*

    o *servlet-api.jar*

    o *sip-servlets-spec-4.0.21.jar*

    o *websocket-api.jar*

Creating the Build Environment

The Dialogic JSR 309 Verification Application source project comes with all necessary components to compile and build application war file.

Prepare the Eclipse Workspace
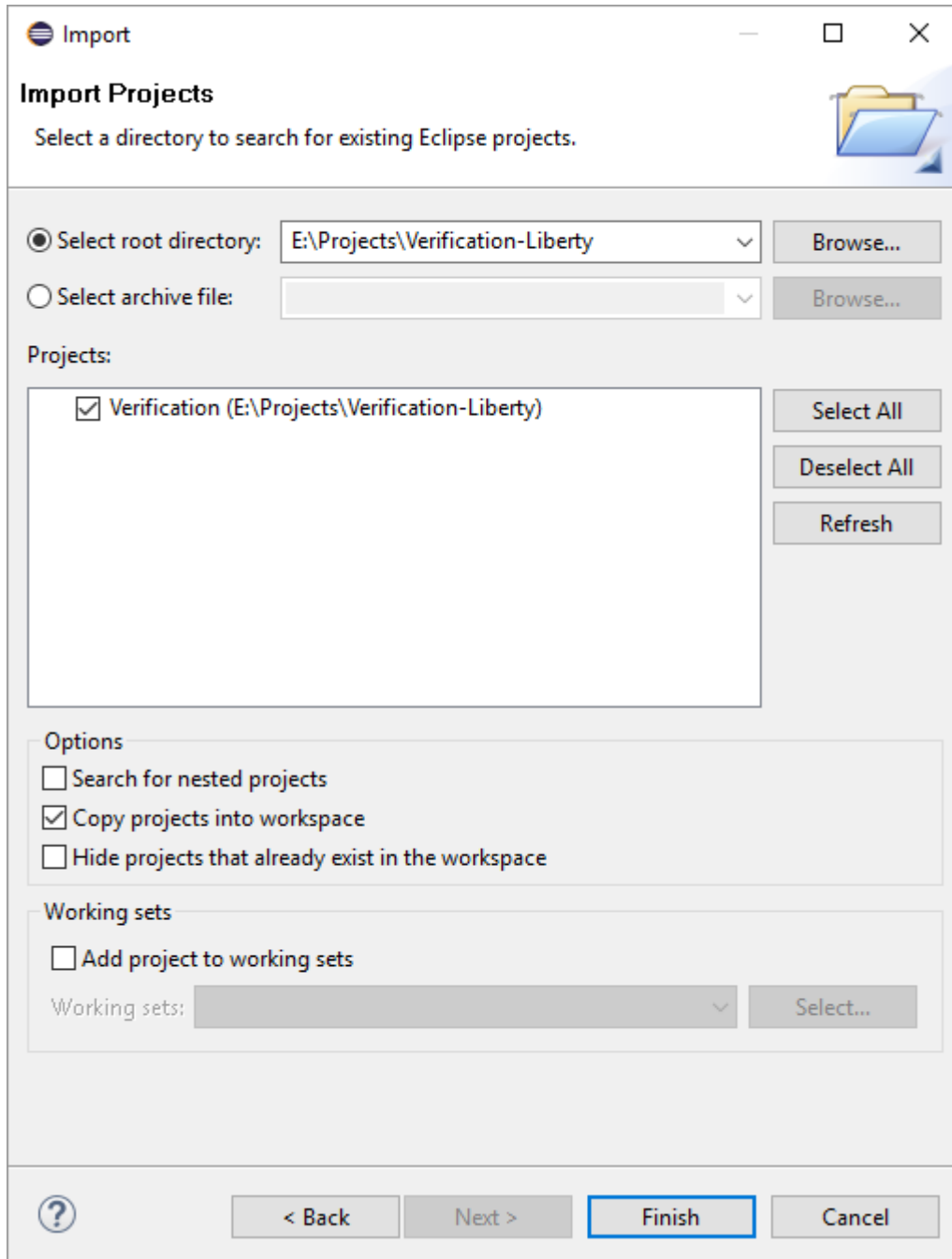
Follow these steps to prepare the Eclipse worskspace:

1. From the distribution package, copy the *DlgcJSR309/application/Project/dlgc_sample_demo.zip* file and extract it to a known location on the system.

2. Verify that the *Project/lib/project/3rdParty* folder contains all required third-party JAR files. Now, we just need to

3. Copy the required Application Server platform specific libraries into the *Project/lib/project/AS* directory.

4. Open Eclipse IDE and click **File > Import**.

5. Select **Existing Project into Workspace**, and then click **Next**.
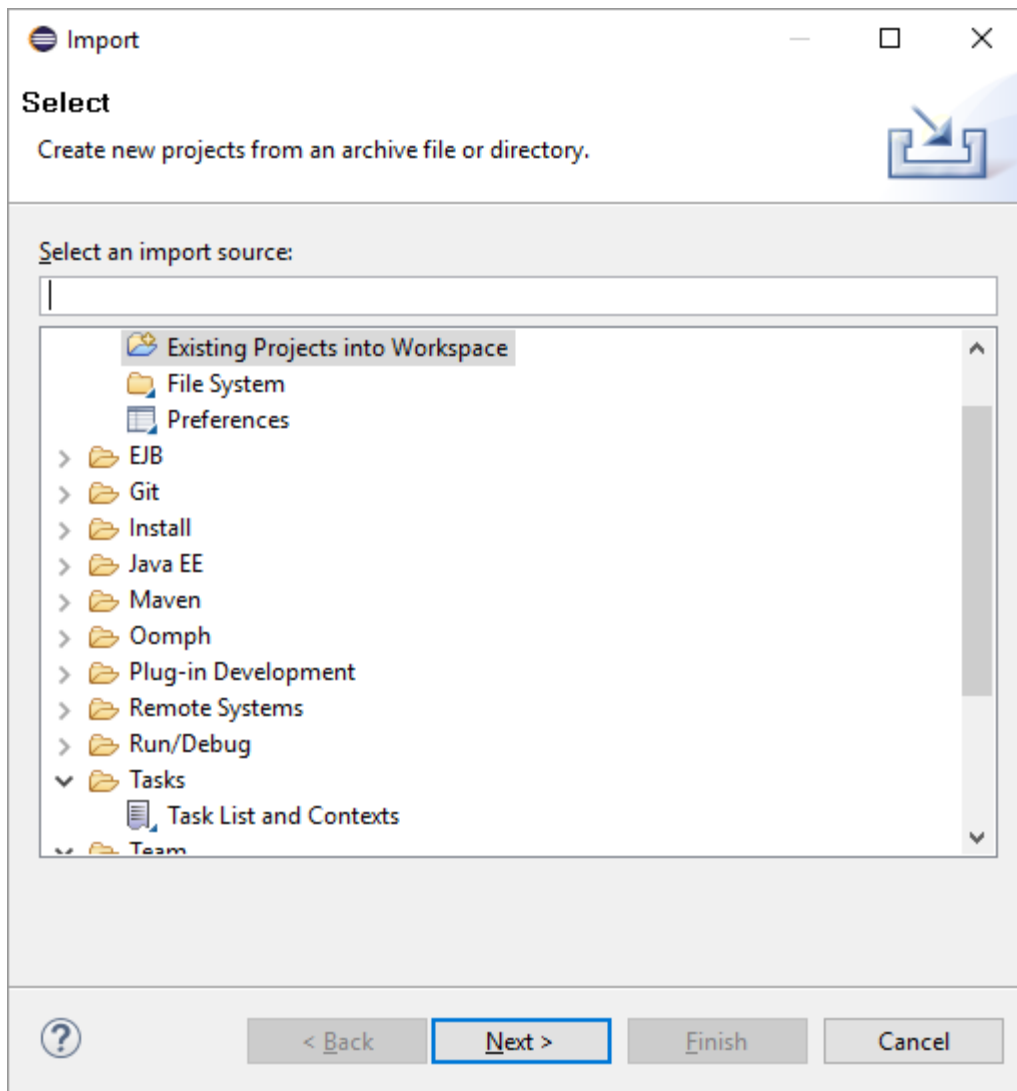
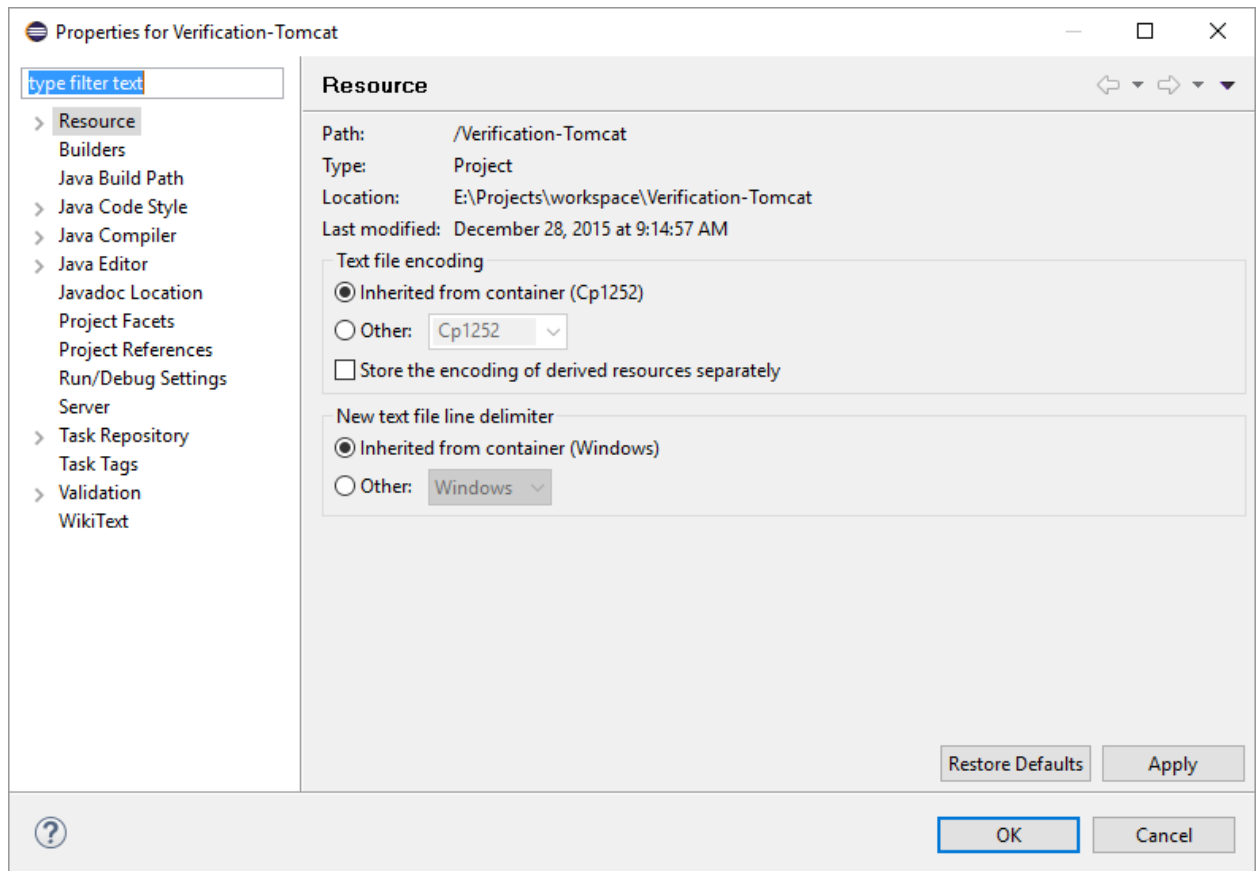6. Click **Browse** and navigate to the *Project* directory on the system.

7. Click **Finish**. The Project has been imported to the Eclipse workspace.
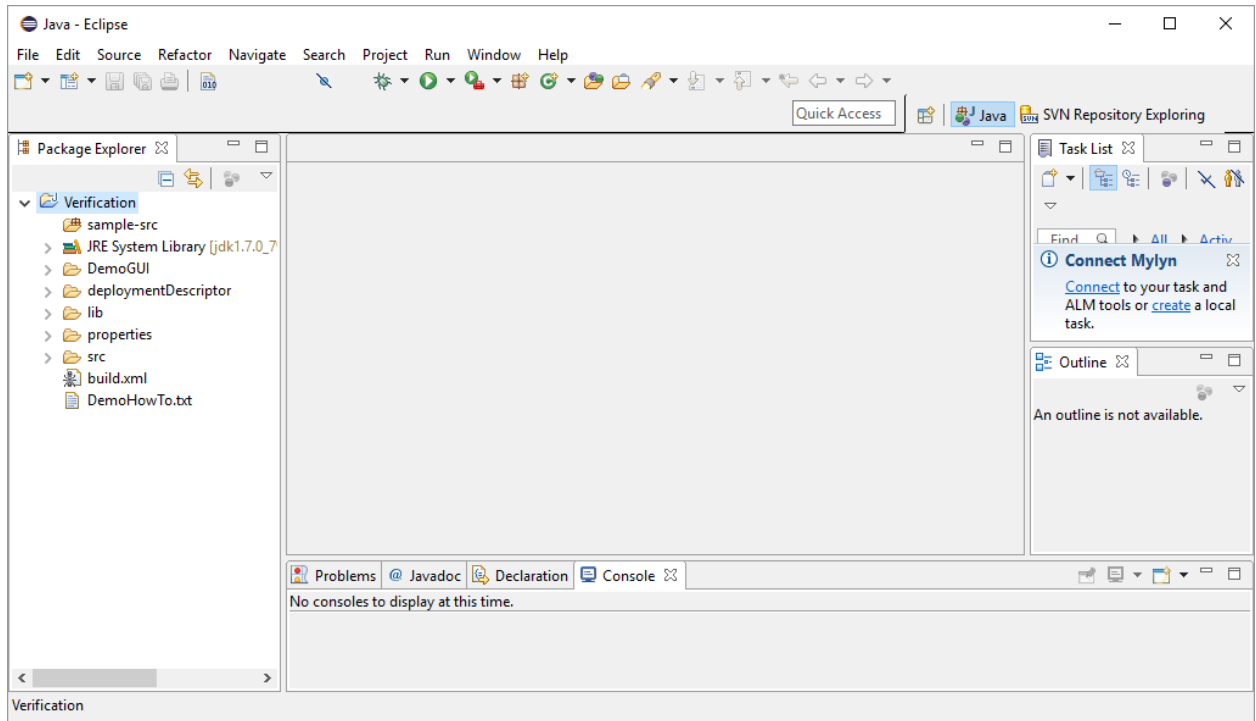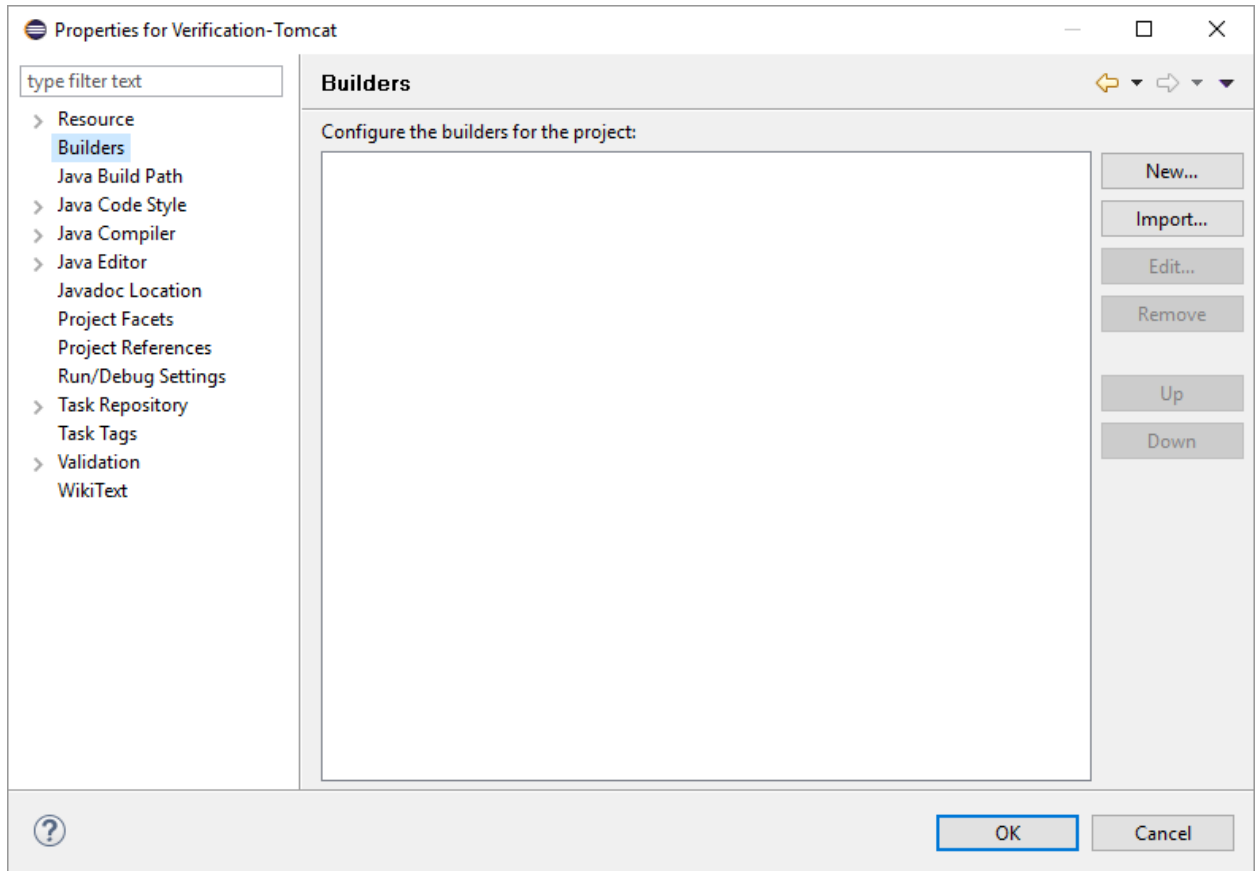
Configure the Application

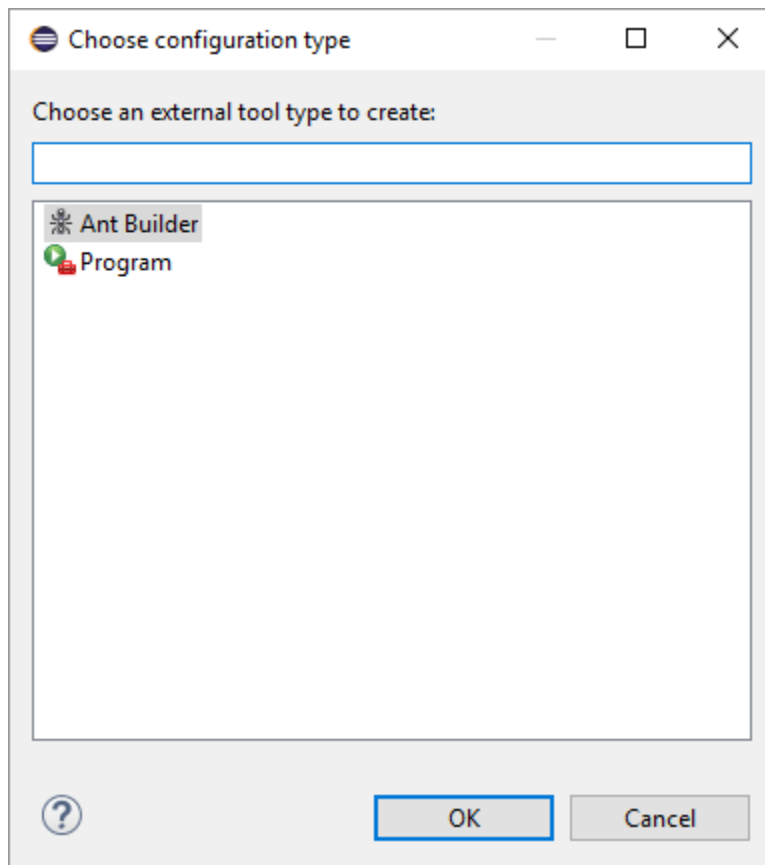To configure the application, edit the following settings:

1. Right-click the **Verification** project folder (for this example) and select **Properties**.

2. Click **Builders** and then click **New**.

3. Click **Ant Builder** and click **OK**.

4. Select any name for this ANT build process. "New_Builder" is the default.

5. In the **Buildfile** section, click **Browse Workspace**.

6. Click the project folder on the left window, select *build.xml* in the right window, and then click **OK**.
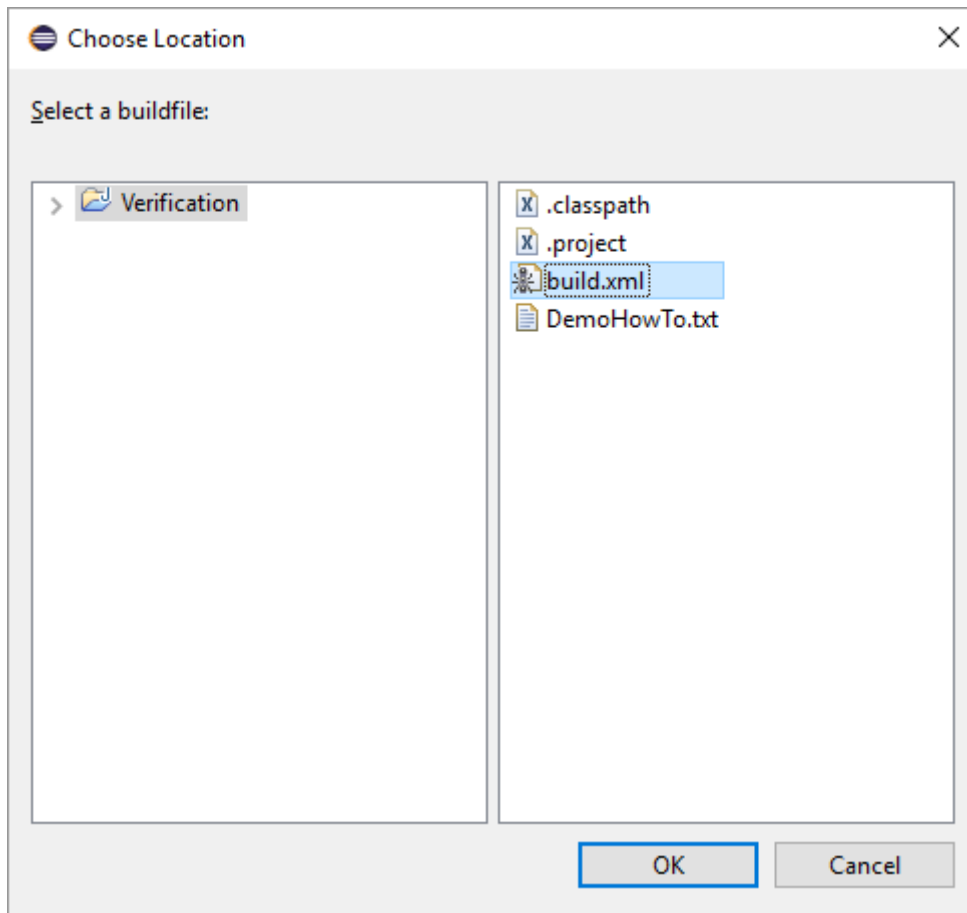
7. In the **Base Directory** section, click **Browse Workspace**.
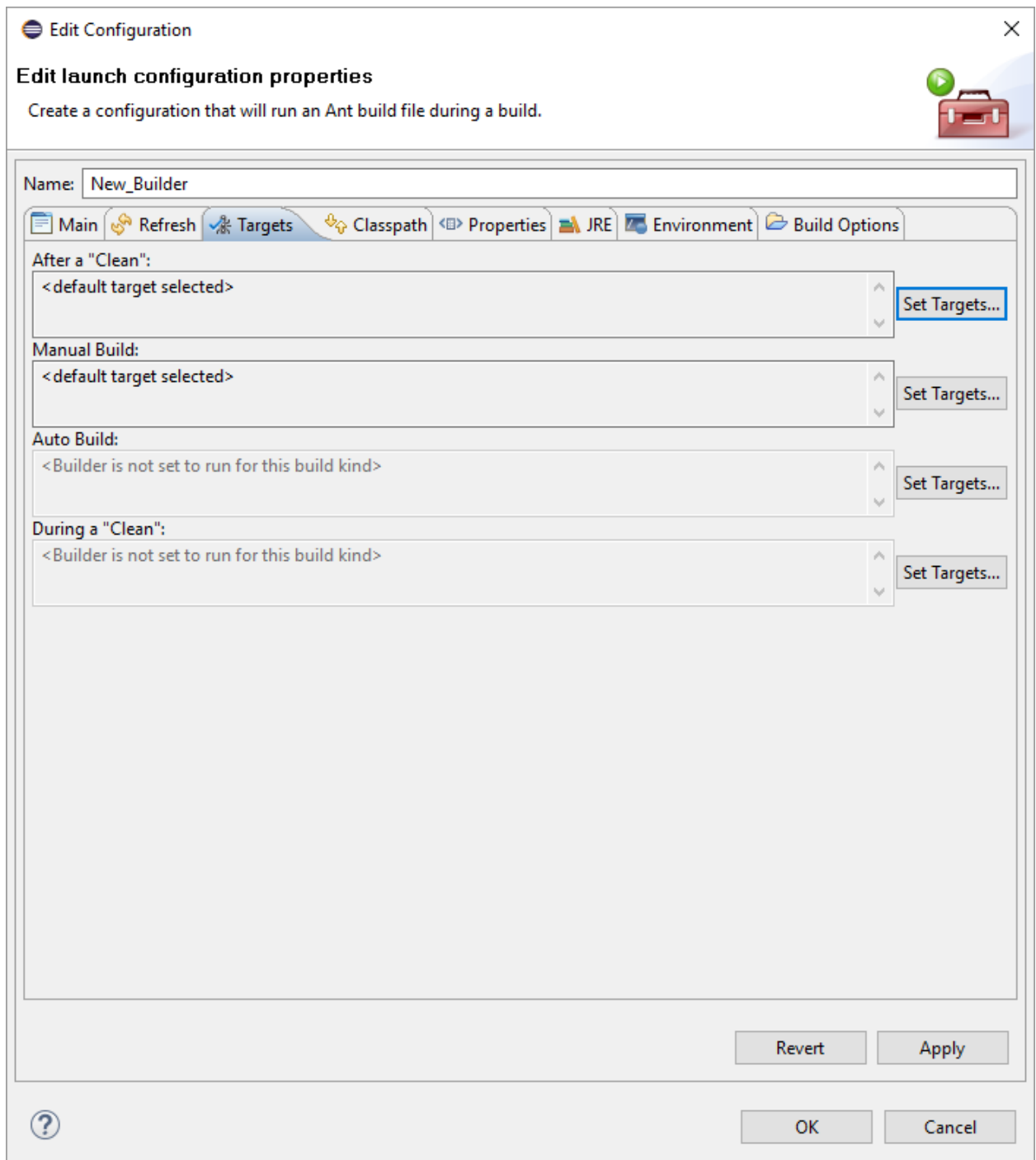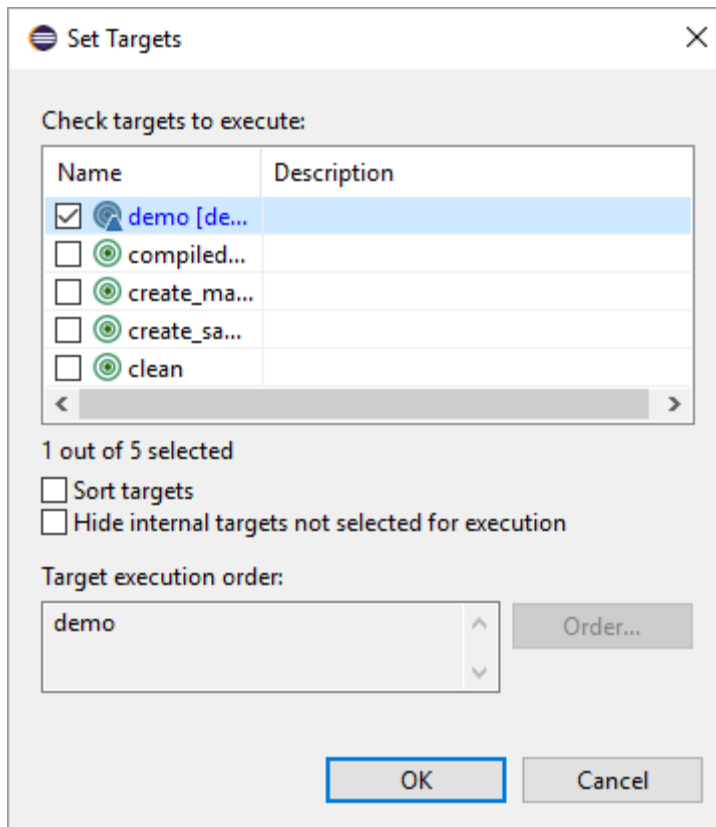
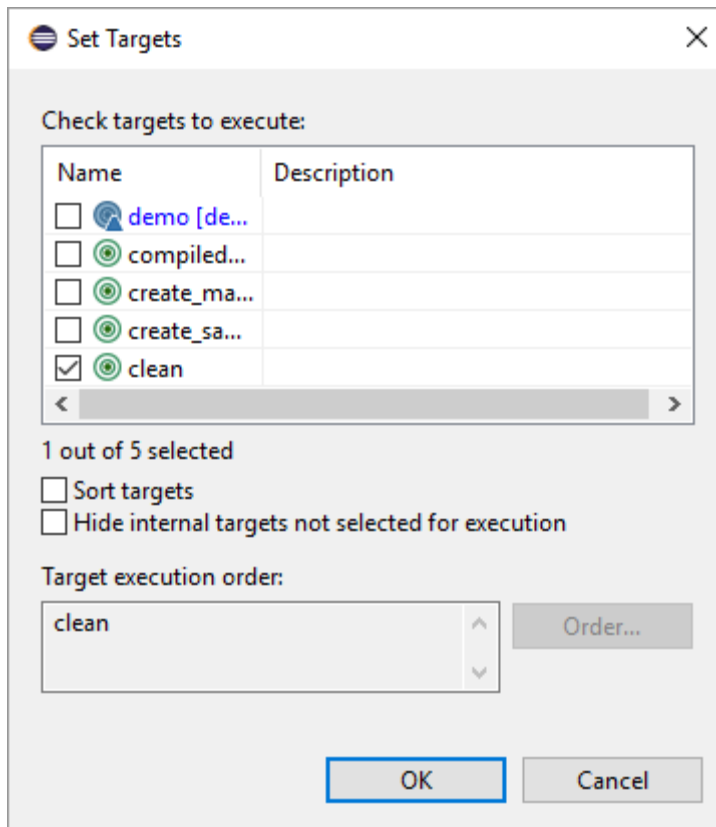8. Select the project directory, and then click **OK**.
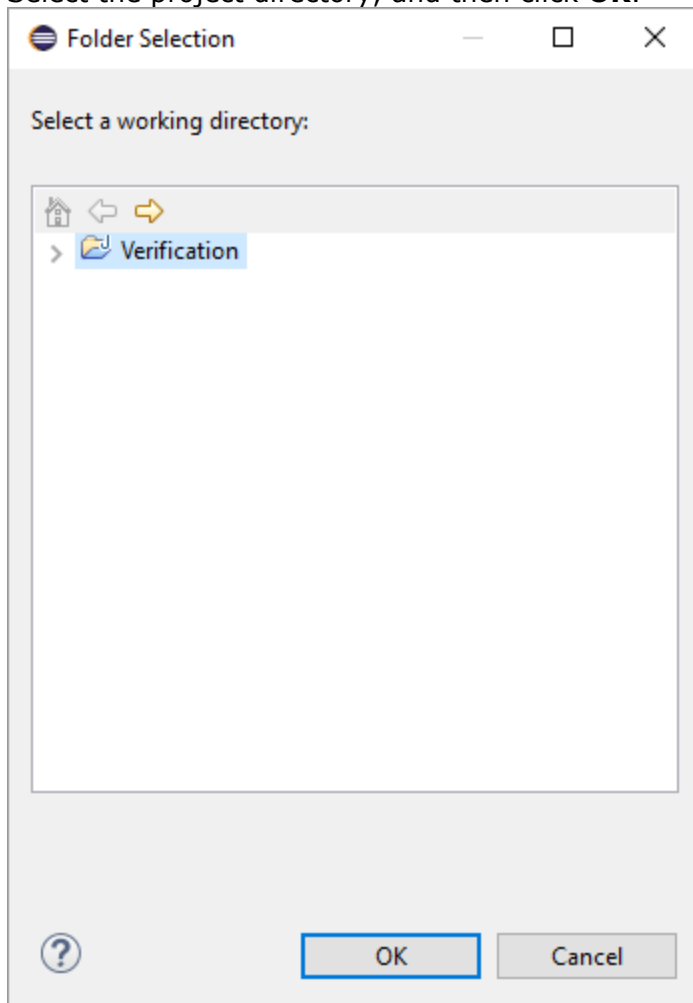
9. Click the **Targets** tab.

10. For **After a "Clean", Manual Build**, and **Auto Build**, make sure that default target is selected. Do that by clicking **Set Targets** for each corresponding section. Make sure that **demo [default]** is selected, and then click **OK**.

11. For the **during a "Clean"** section, click **Set Targets**, make sure that **clean** is the only target that is selected, and then click **OK**.

12. Click the **JRE** tab.
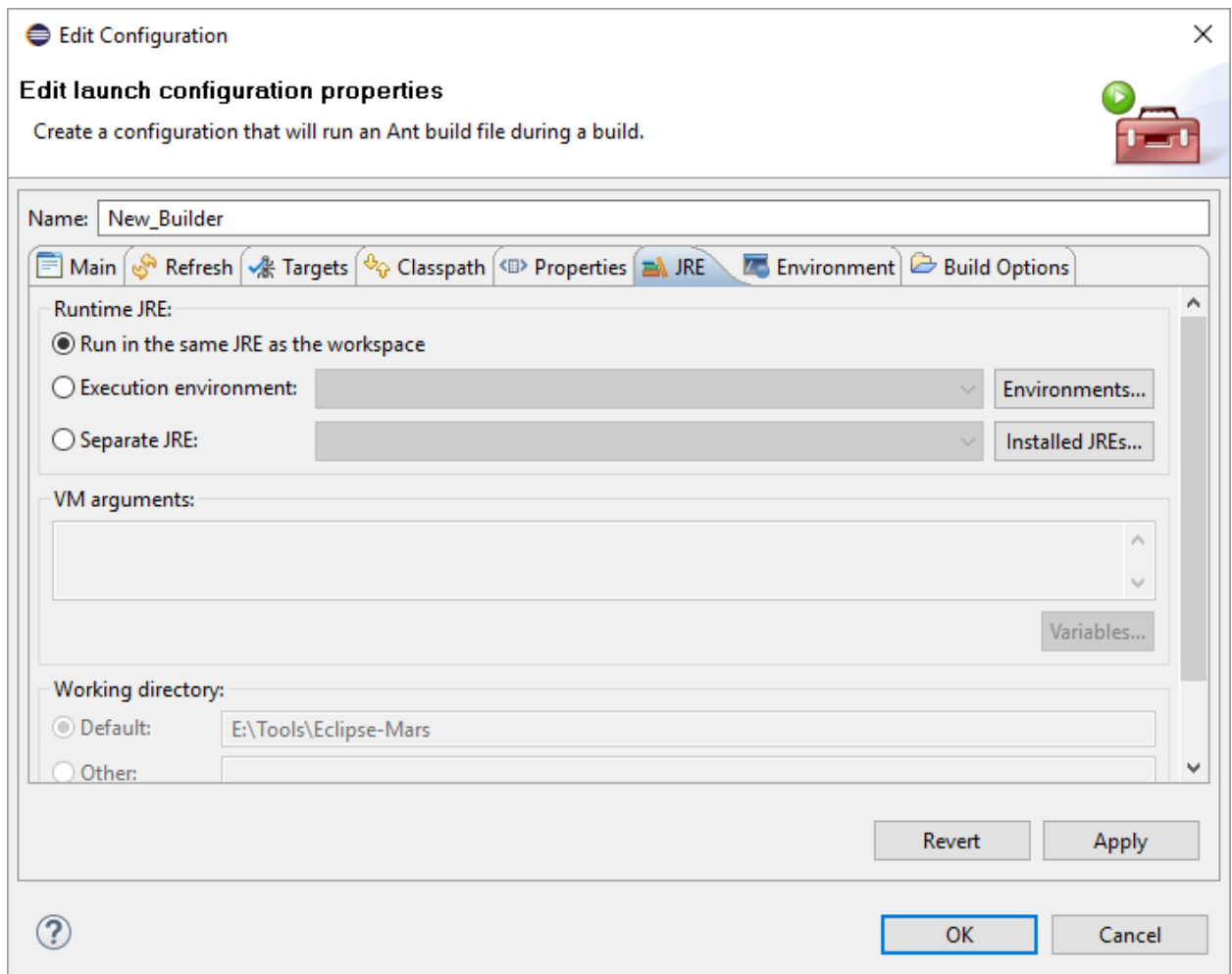
13. Under **Separate JRE** drop-down, select the appropriate JDK as required by the platform. If not present there, click **Installed JREs** to link a project to correct JDK. Make sure that appropriate JDK is selected, and then click **OK**. In this example, the appropriate JDK is jdk1.7.0_79.

14. Click **Java Build Path.**

15. Verify that all jar files are correctly referenced. If there are no project jar files referenced already, click **Add JARs**, select all jar files from within *lib/project/* folders, and then click **OK**.
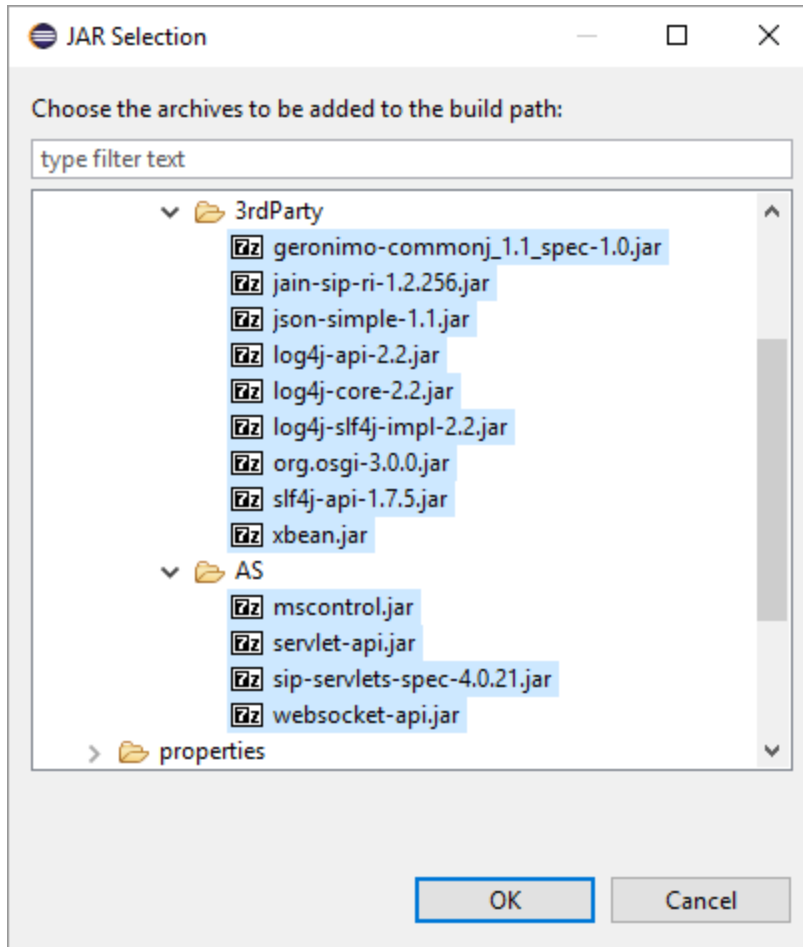
16. Once done, click **OK.**
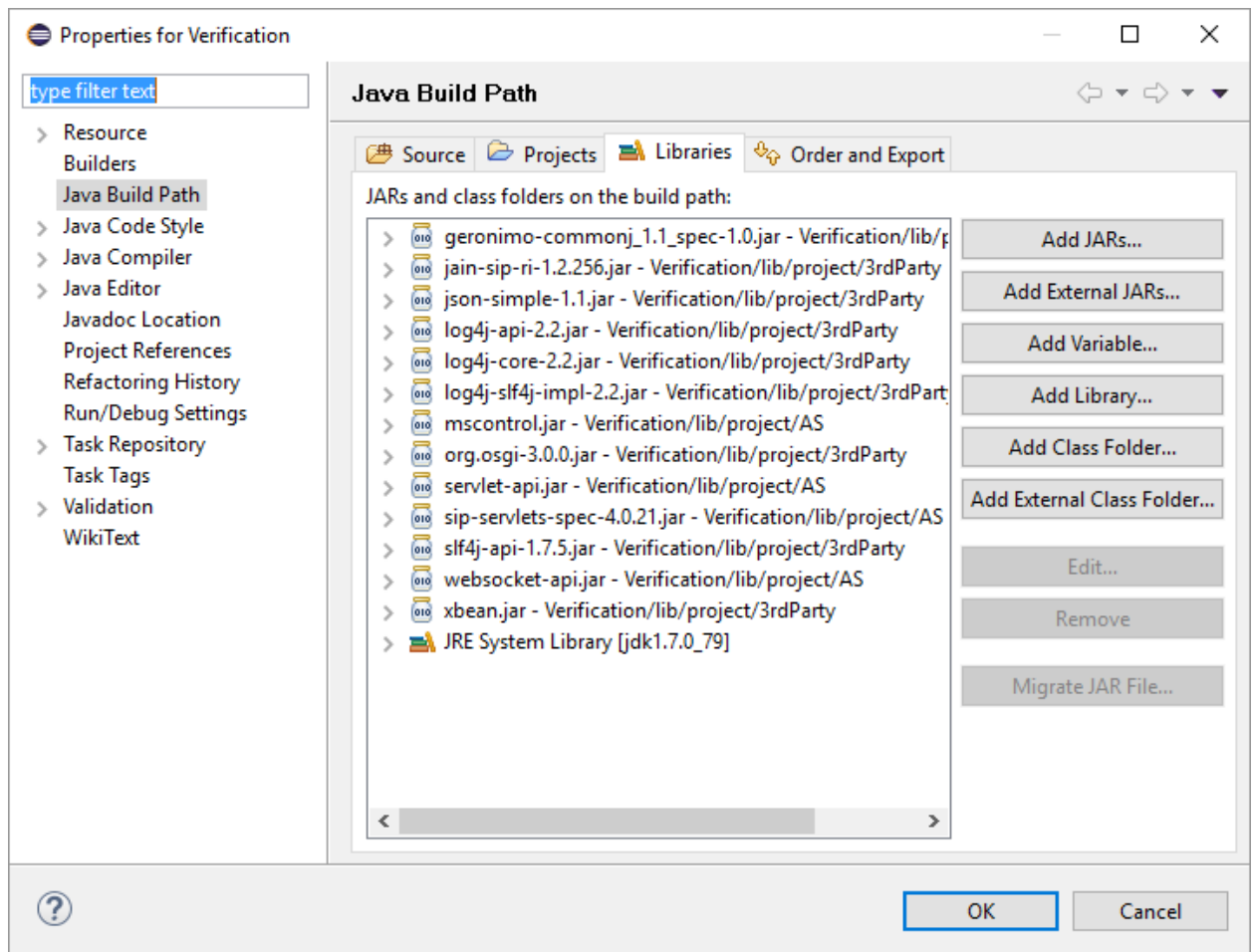
Building the Project

After a successful project installation and configuration, a project can be built. In Eclipse, select the newly created project. In the **Project** menu, click **Build All**. Successful build content will be shown in the **Console** view in Eclipse as follows.

The newly built application WAR file will be located under the *demo_app* directory named

Dialogic-Samples.war. In order to deploy this application, follow the same deployment instructions as described in Installation and Configuration of the Dialogic JSR 309 Connector Verification Application.

Configuring Eclipse Project and Liberty Application Server Deployed Application for Remote Debugging

In order to connect the newly created project to the deployed WAR file in the Application Server for debugging purposes, developers need to follow two simple steps:

- Configure Application Server platform for remote debugging.
- Have Eclipse successfully build the Dialogic JSR 309 Connector Demo Application WAR file and deploy it in the desired Application Server platform. Refer to Deploy Dialogic JSR 309 Connector Demo Application.

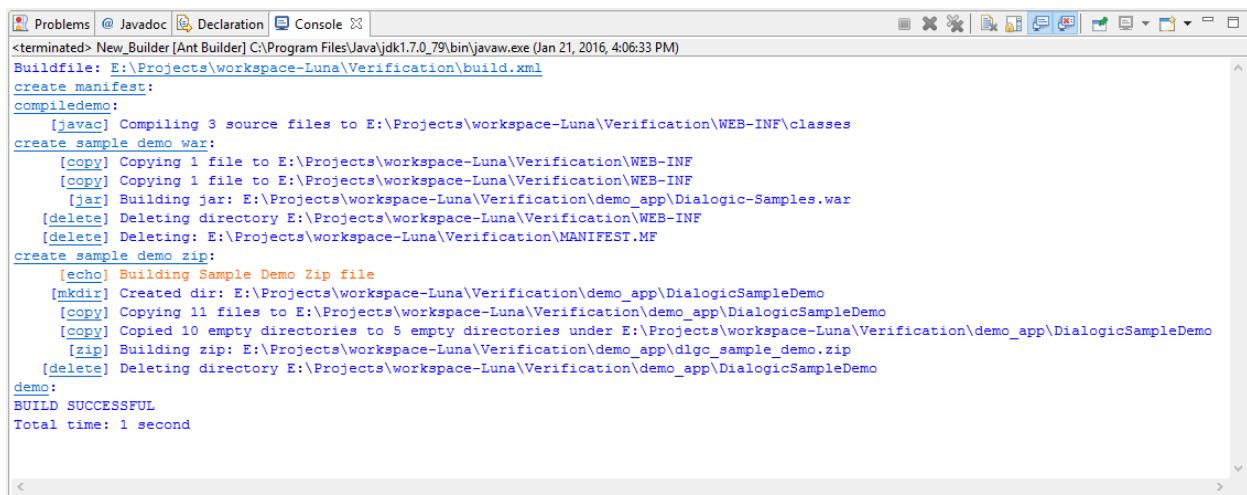## Configuring the Application Server Platform for Remote Debugging

Configure the Application Server platform for remote debugging as follows:

1. Go to the following directory:

```
<Home Dir>/wls/bin/
```

2. Stop Application Server.

3. At the command prompt, execute the following command:

```
export WLP_DEBUG_ADDRESS=8787
```

4. Start application server in debug mode:

```
<Home Dir>/wls/bin/./server debug
```

Note: The socket address specified above is 8787 but any port of choice can be used. Any port used needs to be enabled in a firewall in order to allow communication through it.

Note: The server will start once the remote debugging connection is established. See the next section to set up Eclipse for remote debugging.

Configuring the Eclipse Project for Remote Debugging

To configure the existing and working Dialogic JSR 309 Connector project, the remote debugging section needs to be configured. In Eclipse, go to the **Run** menu and click **Debug Configurations**.
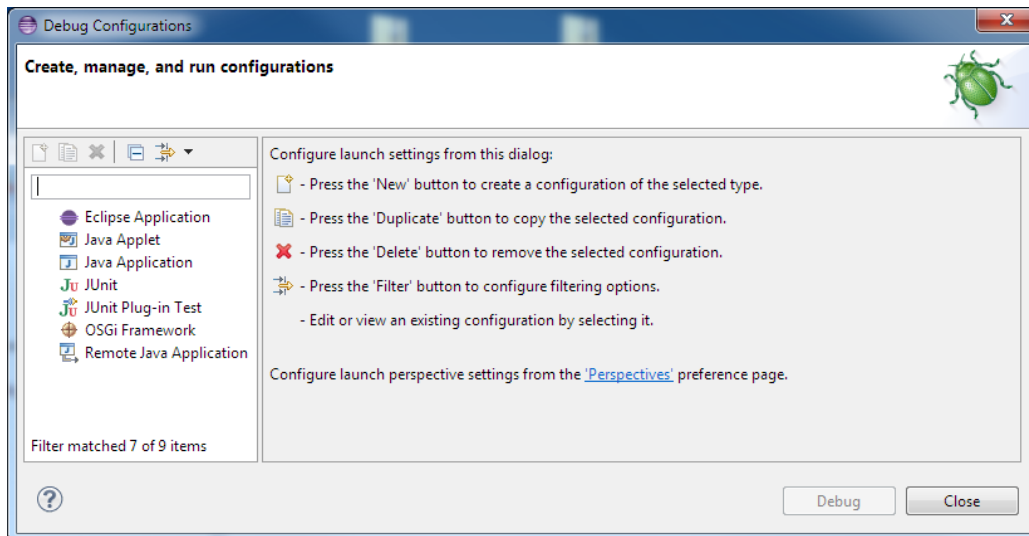
1. In the **Debug Configurations** window, double-click **Remote Java Application**.
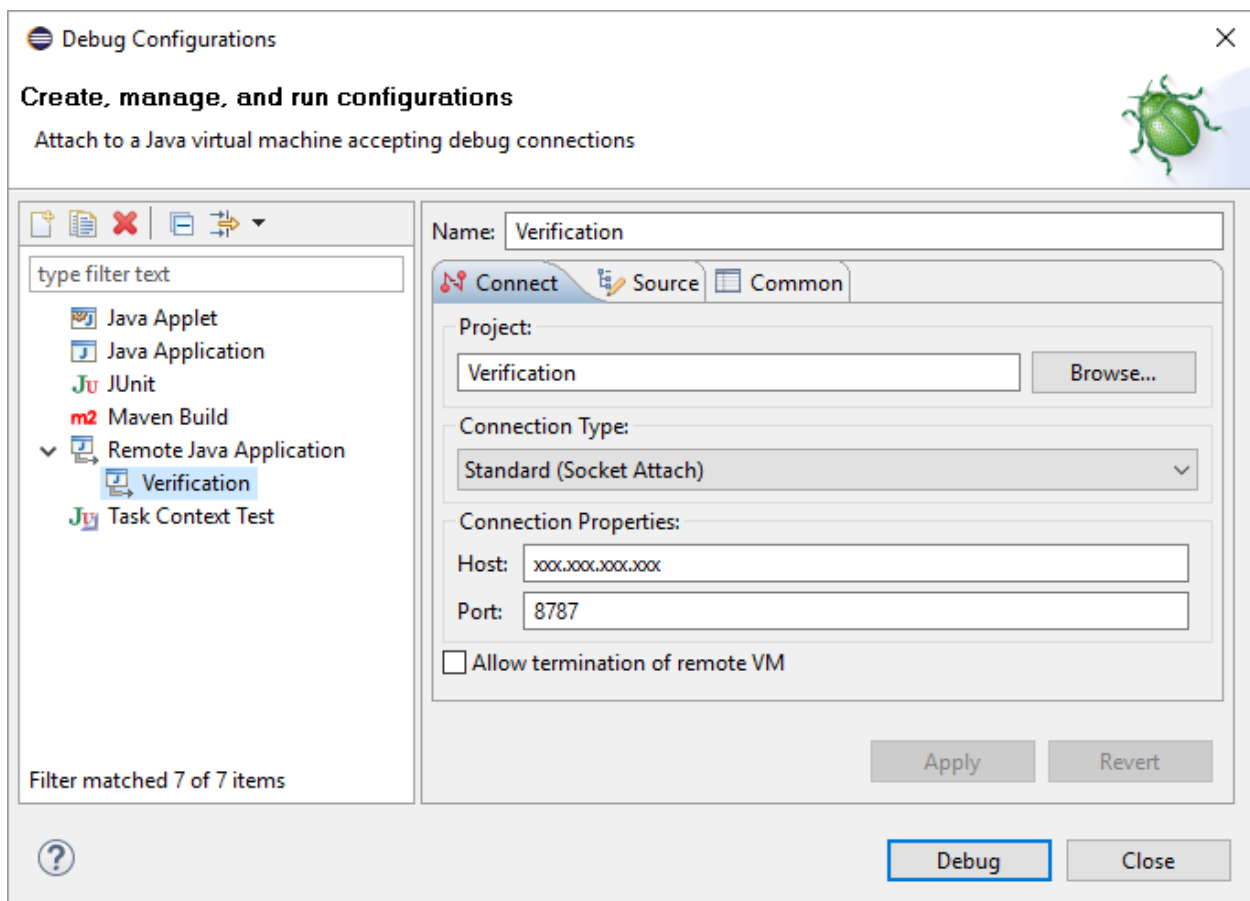
2. In the **Connection Properties** section, specify the **Host** address of the IBM Liberty Server running the deployed application. Specify the debug **Port** as defined when configuring Liberty for remote debugging. Then, click **Apply**.

3. Click **Debug**. The Application Server needs to be running at this point. If not, Eclipse will report a connection error message. If the AS is running but Eclipse is still reporting a connection error, this could be due to either a port mismatch between Eclipse and AS firewall settings and are not allowing the specified port to be used, or there was a port conflict.
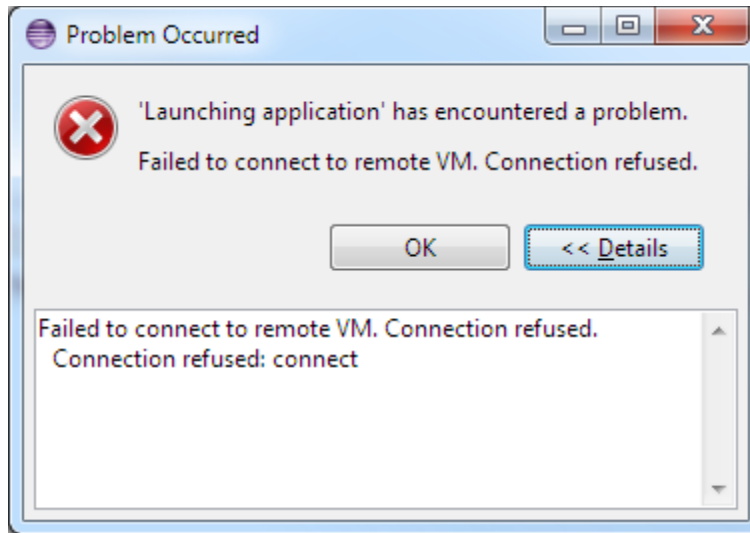

4. Open **debug perspective** in Eclipse (**Windows > Open Perspective > Debug**).

If nothing shows under the **Debug** section of the **debug perspective**, then a connection to the AS has not been established. To connect/reconnect, click the debug icon on the toolbar and choose the newly created **remote debugging configuration**.
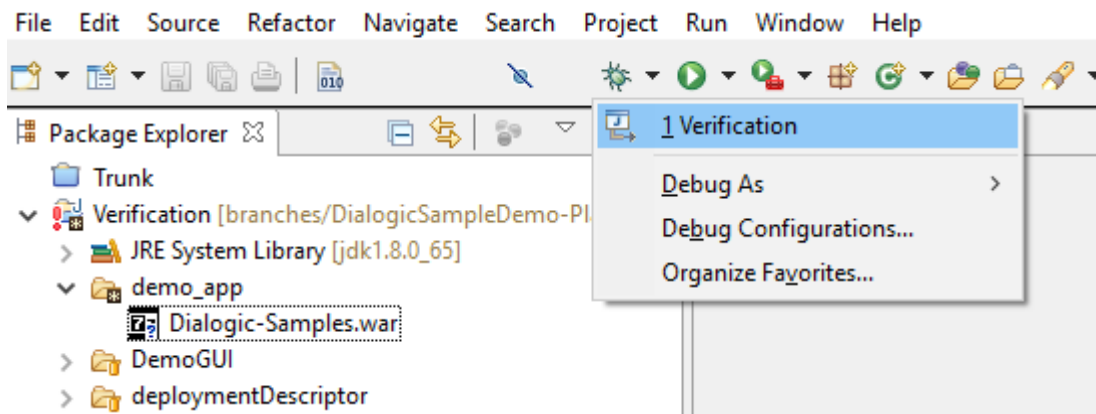
Once **remote debugging configuration** is selected and a connection is established, the content of the **Debug** window will show running threads as follows. The Eclipse project is now connected to the build application that is deployed in the IBM Liberty Application Server.
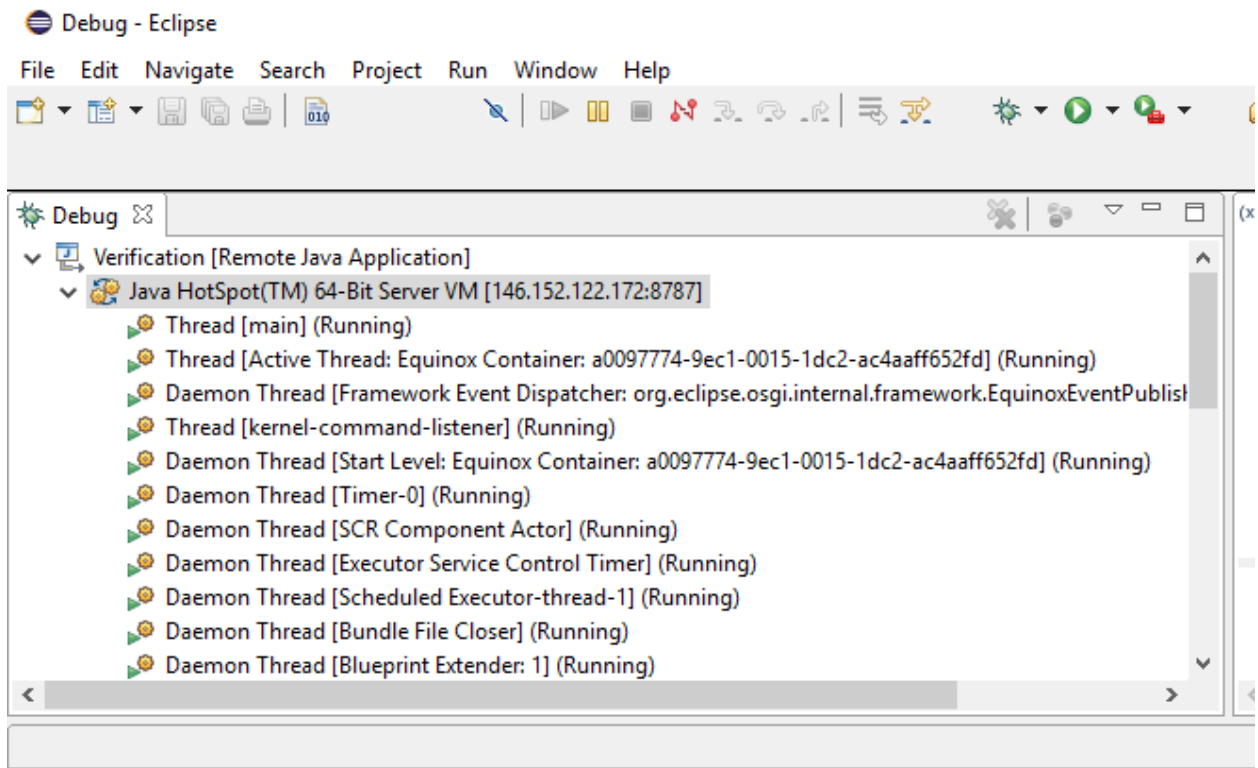
7. Appendix A: Dialogic JSR 309 Connector Environment Setup

Firewall Configuration

Several ports must be allowed to go through the firewall. Refer to the IBM Liberty documentation for specific ports used. For development purposes, firewall can be disabled for quicker setup.

IBM Liberty Installation and Configuration

Install and configure IBM Liberty as follows:

1. Download the latest Liberty beta, which can be obtained from wasdev in the following link:

   https://developer.ibm.com/wasdev/downloads/liberty-profile-beta/

2. After the Liberty beta is downloaded, choose the location where the Liberty server will be placed and extract the zip file.

3. In the *wlp\bin* directory, open a command prompt and run the following commands to install the required features.

   - featureManager install sipservlet-1.1
   - featureManager install mediaservercontrol-1.0

   These features are required to run the sample code given. The feature that will expose the JSR309 API to applications is mediaServerControl-1.0.

4. Now, a server needs to be created. Move to *wlp/bin* and run the following:

   ```
   server create <serverName>
   ```

   Note: This command will create a Liberty server with the specified name.

   Note: If no serverName is specified, a server with "defaultServer" name is created.

Afterthe server is created, a new directory should be present where the server configurations are available. This directory will be located on wlp/usr/servers/<serverName>.

IBM Liberty Startup

The Application Server is ready to run.

1. Go to the following location:

   ```
   <Home Dir>/wls/bin
   ```

2. Execute the following command:

   ```
   ./server start
   ```

To stop the service, execute the following command:

```
./server stop
```

8. Appendix B: Updating the Dialogic JSR 309 Connector User Feature

To update Dialogic JSR 309 Connector User Feature, perform the following procedure:

1. Stop the IBM Liberty server.

2. Remove the existing user feature files that were created as part of original user feature installation:

   ```
   <Home Dir>/wls/usr/extension/lib/com.vendor.dialogic.javax.media.mscontrol.
   LIBERTY.BUNDLE.snapshot_x.x.x.jar
   ```

   ```
   <Home Dir>/wls/usr/extension/lib/features/com.vendor.dialogic.javax.media.mscontrol.
   LIBERTY.snapshot.mf
   ```

3. Place a new Dialogic JSR 309 Connector User Feature on the system in a known location (for example, <Home Dir>).

4. Go into the following directory:

   ```
   <Home Dir>/wls/bin
   ```

5. Run the feature installation command:

   ```
   ./featureManager install /<Home
   Dir>/com.vendor.dialogic.javax.media.mscontrol.LIBERTY.snapshot_5.0.1.esa
   ```

6. Start the IBM Liberty server.