Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 5.2

Installation and Configuration Guide with TeleStax JBoss Application Server

March 2016 Rev 1.0

www.dialogic.com

Copyright and Legal Notice

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see http://www.dialogic.com/company/terms-of-use.aspx for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, PowerVille, PowerNova, MSaaS, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Table of Contents

1. Dialogic JSR 309 Connector Requirements 5

2. Contents of the Distribution 6

3. Installation and Configuration 8

Preparing the J2EE Converged Application Server 8
Installing the Dialogic JSR 309 Connector 8
Configure the Application Server Platform 8
Deploying the Verification Application Using the Dialogic JSR 309 Connector 11
Configure the dlgc_sample_demo.properties File 11
Copy All JAVA Library (JAR) Files 12
Deploy the Dialogic JSR 309 Verification Application 13
Configuring the PowerMedia XMS Media File 18
Running the Dialogic JSR 309 Verification Application 20
Configure Platform SIP Routing Using SIP Servlets Management 21
Dial in to the Dialogic JSR 309 Verification Demo 22

4. Dialogic JSR 309 Verification Application 23

About 23
The Details 23
Application WAR File Content 23
Application Initialization Steps 24
Application Steps to Initialize the Dialogic JSR 309 Connector 25

5. Troubleshooting 27

Logging 27
Dialogic JSR 309 Connector and Verification Application Troubleshooting 28
SIP Errors 29

6. Building and Debugging Sample Demos in Eclipse IDE 30

Prerequisites 30
Creating the Build Environment 30
Prepare the Eclipse Workspace 30
Configure the Application 34
Building the Project 42
Configuring Eclipse Project and TeleStax Application Server Deployed Application for Remote Debugging 43
Configuring the Application Server Platform for Remote Debugging 43
Eclipse Project Configuration for Remote Debugging 44

7. Appendix A: Dialogic JSR 309 Connector Environment Setup 47

Installing and Configuring the TeleStax JBoss Application Server 47
Preinstallation Setup 47
Firewall Configuration 48
TeleStax Installation 49
TeleStax Configuration 49
TeleStax Startup 50
TeleStax Verification 51

8. Appendix B: Updating the Dialogic JSR 309 Connector 53

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 TeleStax JBoss platform for development and testing.

  The Dialogic JSR 309 Connector has been tested with the following JBoss versions of TeleStax Application Servers:

  - TeleStax Mobicents JBoss AS:

    Java 1.7 based:

    mss-3.1.633-jboss-as-7.2.0.Final
  - TeleStax Mobicents WildFly AS (Note: WildFly is BETA):

    Java 1.8 based:

    mss-4.0.21-wildfly-8.2.1.Final
  - TeleStax TelScale JBoss AS:

    Java 1.7 based:

    TelScale-SIP-Servlets-7.0.3.329-jboss-as-7.2.0.Final

  Note: Refer to www.telestax.com for any additional information about TeleStax Application Server and their licensing.
- A functional PowerMedia XMS Release 3.1 system.

  Note: Refer to Proper Configuration of PowerMedia XMS for additional information.
- SIP phones or soft clients.

  Note: WebRTC is not supported because the JBoss platform does not provide websocket support. However, the BETA version of WildFly does provide WebRTC support.

TeleStax Application Server Platform	Dialogic JSR 309 Connector
mss-3.1.633-jboss-as-7.2.0.Final TelScale-SIP-Servlets-7.0.3.329-jboss-as-7.2.0.Final	dialogic309-5.x.xxxx-jboss.tar
mss-4.0.21-wildfly-8.2.1.Final	dialogic309-5.x.xxxx-wildfly.tar

2. Contents of the Distribution

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

There are a number of platforms that are supported by the Dialogic JSR 309 Connector. Their distributions are based on supported platforms as well as the version of Java they support.

This document covers support for TeleStax open source (Mobicents) and equivalent commercial version (TelScale) distributions that are based on JBoss.

- Java 1.7 based platform:

  dialogic309-M.N.BBBB-jboss.tar
- Java 1.8 based platform:

  dialogic309-M.N.BBBB-wildfly.tar

Where:

- *M* defines a major version number.
- *N* defines a minor version number.
- *BBBB* defines the four-digit build number.

This package contains the following structure:

Dialogic JSR 309 Connector Files	Description
DIR: */DlgcJSR309/application/* CONTENTS: *dlgc_sample_demo.war* *Dialogic.mp4* *Project/*	Directory that contains the 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*. Refer to Dialogic JSR 309 Verification Application for details.
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.

Dialogic JSR 309 Connector Files	Description
DIR: */DlgcJSR309/properties/* CONTENTS: *dlgc_sample_demo.properties* *log4j2.xml*	Directory that contains Verification Application properties files used to set up its configuration and the configuration parameters for the Dialogic JSR 309 Connector 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 Dialogic JSR 309 Connector adds the Media Control API interface to an application running in a J2EE platform. The connector and the application need to be correctly configured on a platform for proper operation.

The following steps are necessary to configure the Dialogic JSR 309 Connector and to verify its proper operation:

1. Preparing the J2EE Converged Application Server
2. Installing the Dialogic JSR 309 Connector
3. Deploying the Verification Application Using the Dialogic JSR 309 Connector
4. Configuring the PowerMedia XMS Media File
5. 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 TeleStax Application Servers as described in Dialogic JSR 309 Connector Requirements. For a quick guide on how to install and configure the desired Application Server (AS) before configuring the Dialogic JSR 309 Connector, refer to Appendix A: Dialogic JSR 309 Connector Environment Setup.

Installing the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is created as a library (JAR file) that can be used by an application. The application is responsible for packaging it as part of the application WAR file.

The Dialogic JSR 309 Connector Verification Application is used to verify Dialogic JSR 309 Connector installation and configuration and to illustrate the necessary steps used in the Dialogic Media Server Control API features. These necessary application level steps are clearly described in Dialogic JSR 309 Verification Application.

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

Configure the Application Server Platform

Place the package TAR file on the TeleStax JBoss Linux server and run the following command:

```
tar -xvf dialogic309-5.x.xxxx-jboss.tar
```

This will create a *DlgcJSR309* directory, which includes all necessary files as described in Contents of the Distribution.

Note: These directories are referenced throughout this document for content required by the Dialogic JSR 309 Connector.

In order to properly configure the Application Server platform, the following steps must be completed:

1. Set Up the Environment Variables
2. Set Up and Configure the Logging Facility

Set Up the Environment Variables

The system environment will need to be modified to define the JAVA_HOME directory as well as the location of the Dialogic Verification Application properties file. There are many ways to accomplish this. The following procedure is an example of one way to accomplish this:

1. Edit the system user's .bashrc file using the following command.

```
vi .bashrc
```

2. Add the following lines marked in RED below. In this example the *mss-3.1.633-jboss-as-7.2.0.Final* Application Server platform is used and the user "jboss" was created on the Linux system. Therefore, the file to be edited will be found in this user's root directory (i.e., */home/jboss*).

```
# .bashrc


### Dialogic additions


export JAVA_HOME=/usr/java/jdk1.7.0_80
export JBOSS_HOME=/home/jboss/mss-3.1.633-jboss-as-7.2.0.Final
export
SAMPLE_PROPERTY_FILE=${JBOSS_HOME}/standalone/configuration/Dialogic/dlgc_sample_demo.pro
perties


### END - Dialogic additions


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


# User specific aliases and functions
```

Note the following about Dialogic additions:

- JAVA_HOME points to a specific Java version. This version has to match the version of JAVA that is supported by the designated platform. For the platform in this example, it is Java 1.7.
- JBOSS_HOME points to a root directory of the installed platform.
- SAMPLE_PROPERTY_FILE points to the configuration properties file used by the Dialogic JSR 309 verification demo.

  Note: This will not be needed if a custom application is being deployed.

In order for above the changes to take effect, a user either needs to log out and log back in or execute the "source" command for the modified .bashrc file:

```
source /home/jboss/.bashrc
```

Set Up and Configure the Logging Facility

The log4j2 logging facility is implemented through five third-party log4j2 library files (JAR files) that need to be placed in the application's WAR file lib directory. From the Dialogic JSR 309 Connector distribution, copy the following five JAR files:

- *log4j-api-2.2.jar*
- *log4j-core-2.2.jar*
- *log4j-slf4j-impl-2.2.jar*
- *slf4j-api-1.7.5.jar*
- *org.osgi-3.0.0.jar*

Place the five JAR files in the platform specific lib folder:

```
${CATALINA_HOME}/lib
```

Edit *standalone.conf*, which is located here:

```
${JBOSS_HOME}/bin/standalone.sh
```

Add the following lines exactly located as shown below:

```
# Uncomment to gather JBoss Modules metrics
#JAVA_OPTS="$JAVA_OPTS -Djboss.modules.metrics=true"


### Dialogic additions
JAVA_OPTS="$JAVA_OPTS -
Dlog4j.configurationFile=${JBOSS_HOME}/standalone/configuration/Dialogic/log4j2.xml"
### END - Dialogic additions
```

From the distribution folder, copy *log4j2.xml* into the following platform directory:

```
${JBOSS_HOME}/standalone/configuration/Dialogic/log4j2.xml"
```

Note: A "Dialogic" directory will need to be created if it does not already exist.

Note the following about the logging facility:

- Due to a known Log4j version 1 Thread Deadlock issue, the Dialogic JSR 309 Connector has been built using log4j2 (version 2). Modification of a platform startup script is required to configure the log4j2 logging facility and to define a reference to its configuration .xml file.
- The *Dialogic.log* file, when generated, is found here:

  ```
  ${JBOSS_HOME}/logs/Dialogic.log
  ```

- Default logging configuration is set to ERROR. Configuration file, *Log4j2.xml,* can be edited if one wishes to change the logging levels.

Note: The *log4j2.xml* file changes go into effect automatically as governed by the configuration parameter in the *log4j2.xml* file. Details of *log4j2.xml* as provided can be found in the Troubleshooting section under Logging.

Deploying the Verification Application Using the Dialogic JSR 309 Connector

The Dialogic JSR 309 Verification Application, like any other application that wants to use J2EE Media Server Control (JSR 309) API, needs to do the following:

1. Take specific steps to correctly initialize itself for the J2EE platform to correctly deploy it.
2. Take specific steps to correctly initialize the Dialogic JSR 309 Connector for its use.

Note: For further details on the application architecture refer to Dialogic JSR 309 Verification Application.

The following steps are necessary to deploy the Dialogic JSR 309 Verification Application:

1. Configure the dlgc_sample_demo.properties File
2. Copy All JAVA Library (JAR) Files
3. Deploy the Dialogic JSR 309 Verification Application

Configure the dlgc_sample_demo.properties File

From the distribution folder, copy the *dlgc_sample_demo.properties* file and place it in the following directory:

```
${JBOSS_HOME}/standalone/configuration/Dialogic
```

Note: The "Dialogic" directory should already exist and contain the *log4j2.xml* file.

From the platform's Dialogic directory, edit the *dlgc_sample_demo.properties* file. Provide the required parameters as illustrated in RED below:

```
# Dialogic JSR 309 Verification Application configuration parameters:
# Dialogic JSR 309 Connector Configuration
        dlgc.jsr309.driver.name=com.dialogic.dlg309
        connector.sip.address=xxx.xxx.xxx.xxx
        connector.sip.port=xxxx
        connector.sip.transport=udp
# Dialogic Media Server Configuration
        mediaserver.sessionTimer.switch=off
        mediaserver.sessionTimer.maxTimeout=120
        mediaserver.sessionTimer.minTimeout=100
        mediaserver.sip.address=xxx.xxx.xxx.xxx
        mediaserver.sip.port=xxxx

# Application runtime parameters:
play.prompt=file:////en_US/verification/Dialogic.mp4
```

Refer to the following information for details on the *dlgc_sample_demo.properties* file:

1. Dialogic JSR 309 Connector Configuration:
    - connector.sip.address – IP address of the SIP interface used by the platform. The connector detects an IP address automatically. However, if there is more than one interface available on the platform, the application will have to be able to set the appropriate IP address for the connector to use. The demo will take this IP address and if different from the automatically discovered IP address, the demo will use it.

- connector.sip.port – Port address of the SIP interface used by the platform. The connector detects a port automatically, which goes with detected IP address. However, if the port address is incorrect, the application will have to set the appropriate one for connector to use. The demo will take this port address, and if different from the automatically discovered port address, the demo will use it.
- connector.sip.transport – Platform SIP transport. Supported values are "udp" or "tcp". Default: "udp".

2. Dialogic Media Server Configuration
    - mediaserver.sip.address – User needs to specify an IP of a Dialogic PowerMedia XMS to be used by the Dialogic JSR 309 Connector.
    - mediaserver.sip.port – User needs to specify a port of a Dialogic PowerMedia XMS 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".

Copy All JAVA Library (JAR) Files

The Dialogic JSR 309 Connector and its Verification Application are dependent on a few third-party JAVA library files.

The Verification Application needs to include all required JAR files inside the application WAR package in the designated lib folder. The following is a view of the directory structure of the Dialogic JSR 309 Verification Application and highlighted is the lib folder where all required JAR files need to be placed:

```
\META-INF\MANIFEST.MF
\WEB-INF\jboss-deployment-structure.xml
\WEB-INF\sip.xml
\WEB-INF\classes\base\ConfigProperty.class
\WEB-INF\classes\play\AsyncPlayer$PlayerEventListener.class
\WEB-INF\classes\play\AsyncPlayer$PlayerJoinEventListener.class
\WEB-INF\classes\play\AsyncPlayer$SdpPortsListener.class
\WEB-INF\classes\play\AsyncPlayer.class
\WEB-INF\lib\Dialogic309-5.#.####-jboss.jar
\WEB-INF\lib\Dialogic309msmltypes-5.#.####-jboss.jar
\WEB-INF\lib\Dialogic309smiltypes-5.#.####-jboss.jar
\WEB-INF\lib\geronimo-commonj_1.1_spec-1.0.jar
\WEB-INF\lib\jain-sip-ri-1.2.256.jar
\WEB-INF\lib\json_simple-1.1.jar
\WEB-INF\lib\jsr173_1.0_api.jar
\WEB-INF\lib\log4j-api-2.2.jar
\WEB-INF\lib\log4j-core-2.2.jar
```

```
\WEB-INF\lib\log4j-slf4j-impl-2.2.jar
\WEB-INF\lib\mscontrol.jar
\WEB-INF\lib\org.osgi-3.0.0.jar
\WEB-INF\lib\slf4j-api-1.7.5.jar
\WEB-INF\lib\xbean.jar
```

Note: mscontrol.jar is a JAVA library JAR file that is distributed by jcp.org. This library is not part of the Dialogic distribution as per its license agreement. This is one of the required libraries. If your platform does have it as part of the install, download it directly from the jcp.org website. The *mscontrol.jar* is imbedded inside the downloadable zip file. The library is located here:

http://download.oracle.com/otndocs/jcp/media_server_control-1.0-fr-eval-oth-JSpec/

Note: Further details on application WAR file content refer to Dialogic JSR 309 Verification Application

Deploy the Dialogic JSR 309 Verification Application

The Dialogic JSR 309 Connector demo application needs to be deployed in the TeleStax Application Server.

There are two ways to deploy an application WAR file in the TeleStax Application Server.

- Deploying via Application Server Web Administration Console
- Deploying directly on the server

Deploying via Application Server Web Administration Console

1. Start the TeleStax Application Server from here:

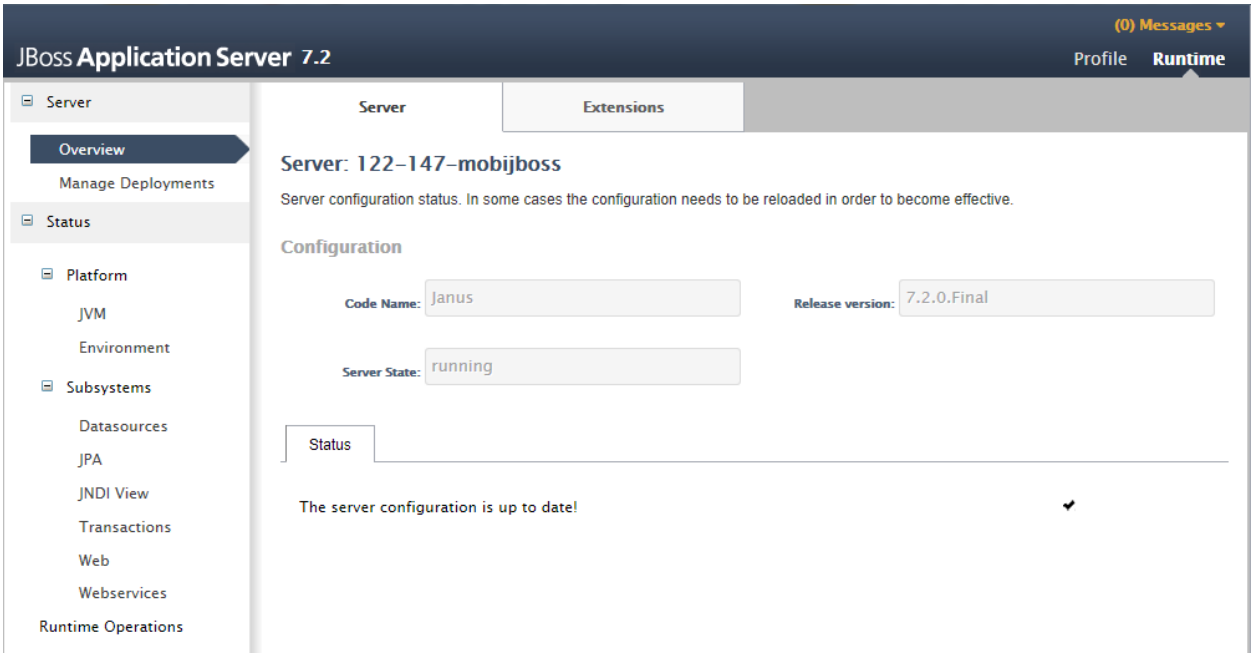
   ```
   ${JBOSS_HOME}/bin
   ```

2. Execute the following script. If the server starts without any errors/exceptions, the application is ready to be deployed.

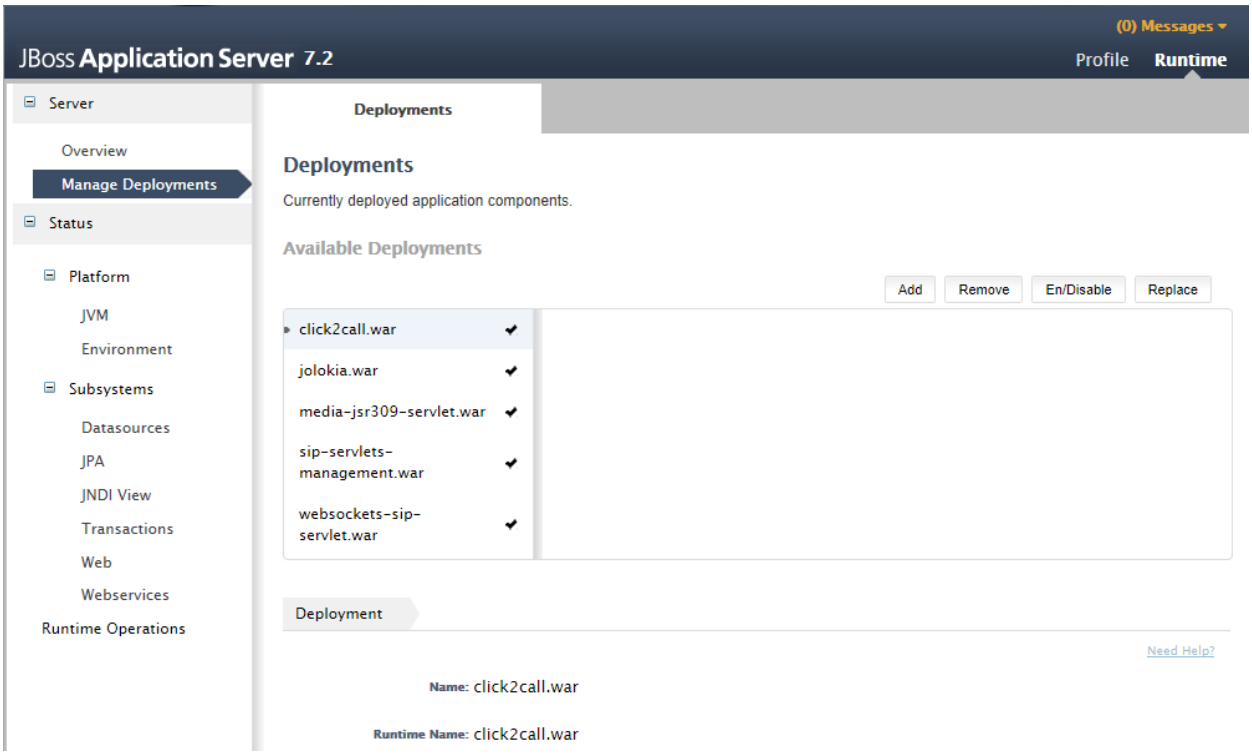
   ```
   ./standalone.sh -c standalone-sip.xml
   ```

3. Navigate to http://<as_ip_address>:8080 and click **Administration Console**.

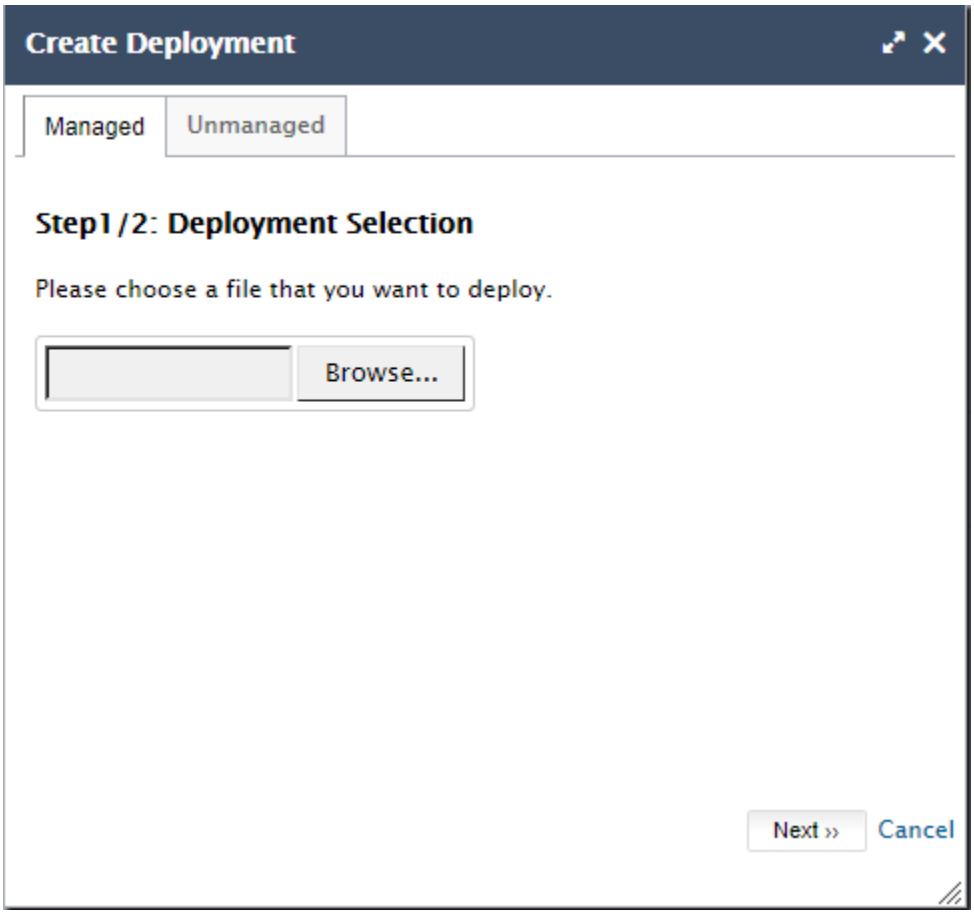
4. Enter the appropriate login credentials.



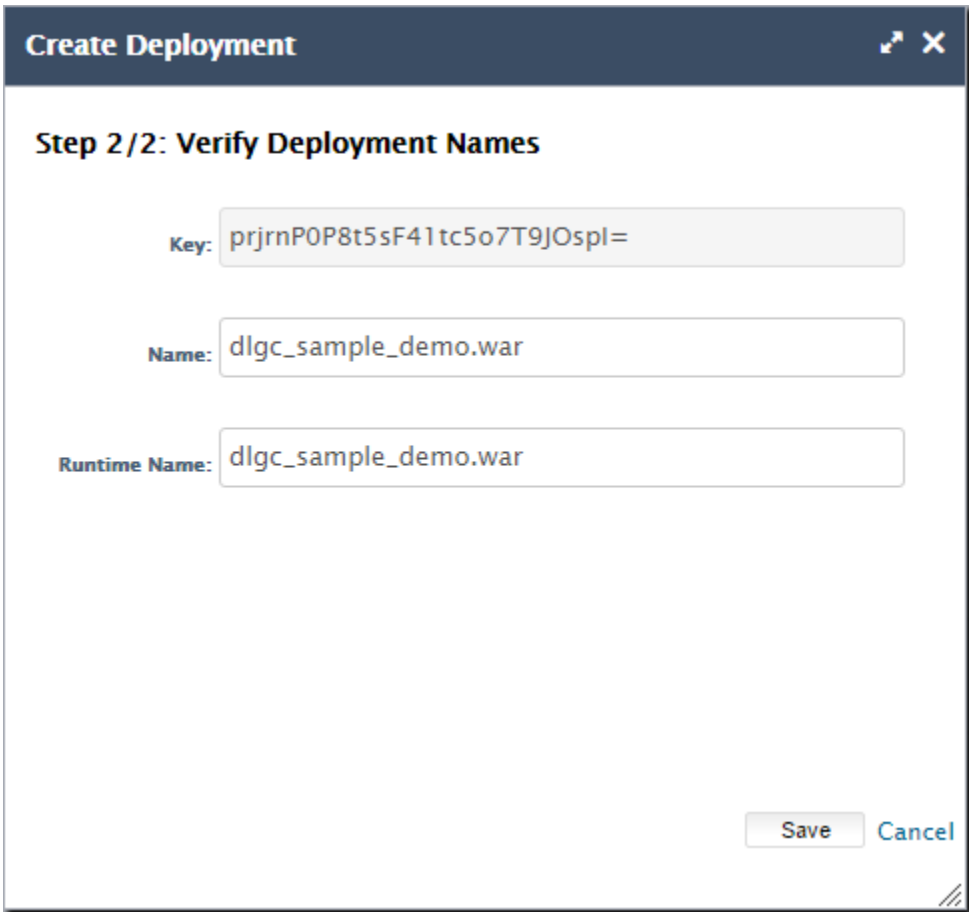
5. Click **Manage Deployments** and then click **Add**.



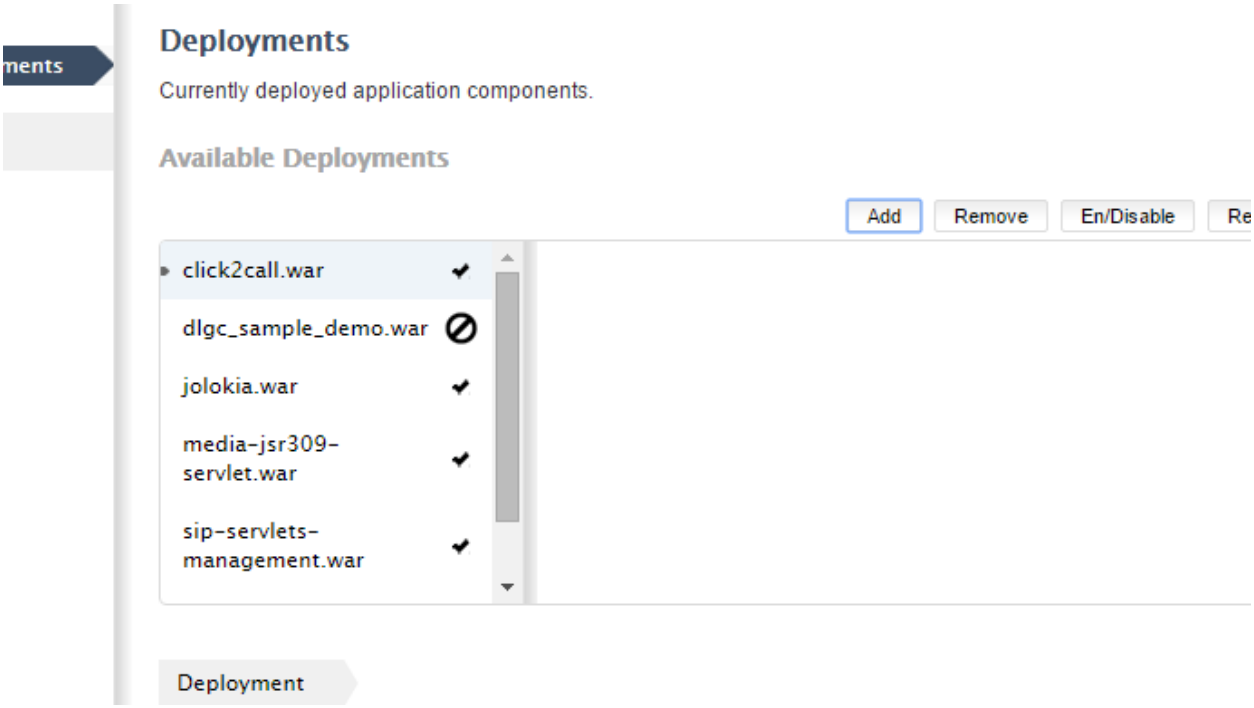
6. Click **Browse** , select the *dlgcmsc_demo.war* file, and then click **Next**.



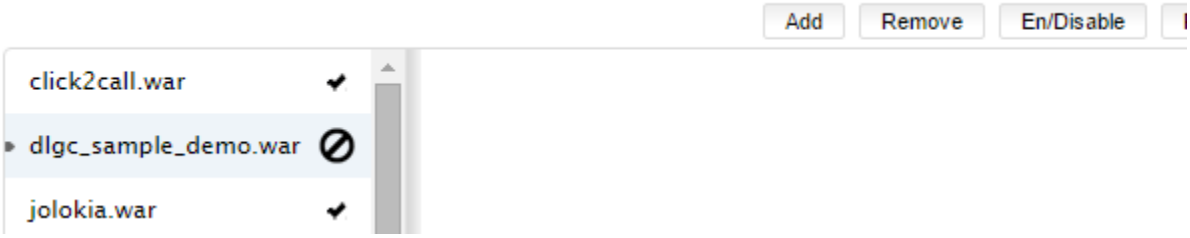
7. Click **Save**.


Create Deployment
Step 2/2: Verify Deployment Names
Key:
prjrnP0P8t5sF41tc5o7T9JOspI=
Name:
dlgc_sample_demo.war
Runtime Name:
dlgc_sample_demo.war
Save
Cancel

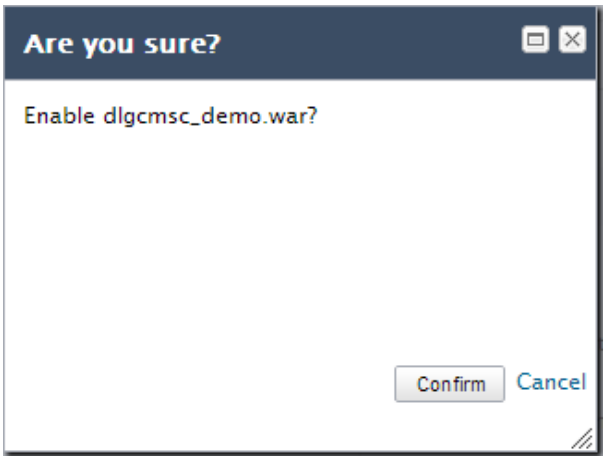
On the **Manage Deployments** page, the newly deployed *dlgc_sample_demo.war* application appears.



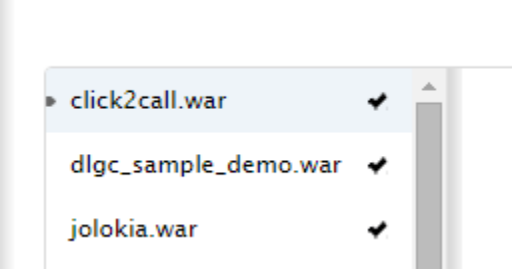
8. To enable the deployed application, highlight it and click **En/Disable**.



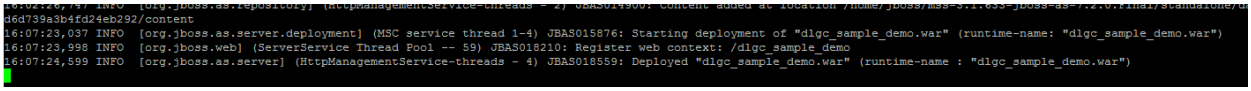
9. Click **Confirm**.



If deployment was competed successfully, the deployed WAR file will have a check icon next to it.



The console window will indicate that *dlgc_sample_demo.war* has been deployed.



Deploying Directly on the Server

Another way to deploy the application is to place the WAR file in the following directory:

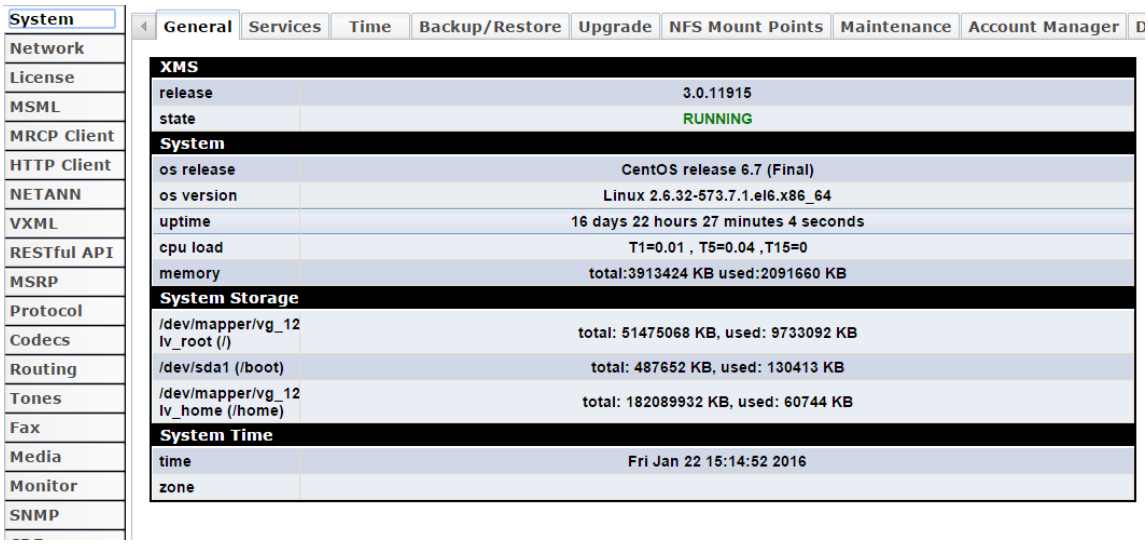
```
${JBOSS_HOME}/standalone/deployments
```

The Application Server is monitoring the deployments directory, so if a new or updated WAR file is detected, it will attempt to start automatically.

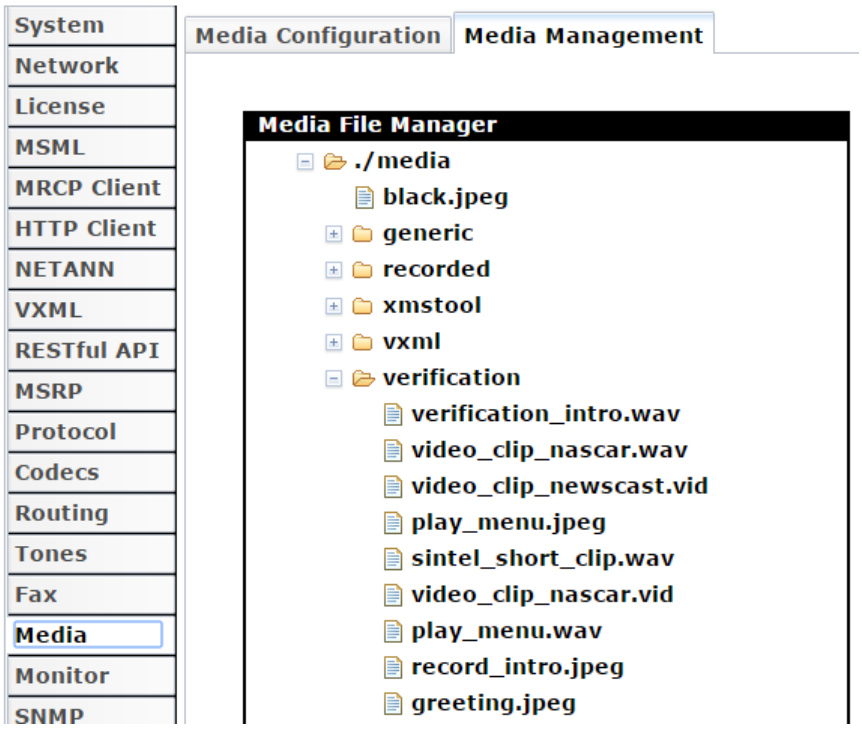
Configuring the PowerMedia XMS Media File

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.

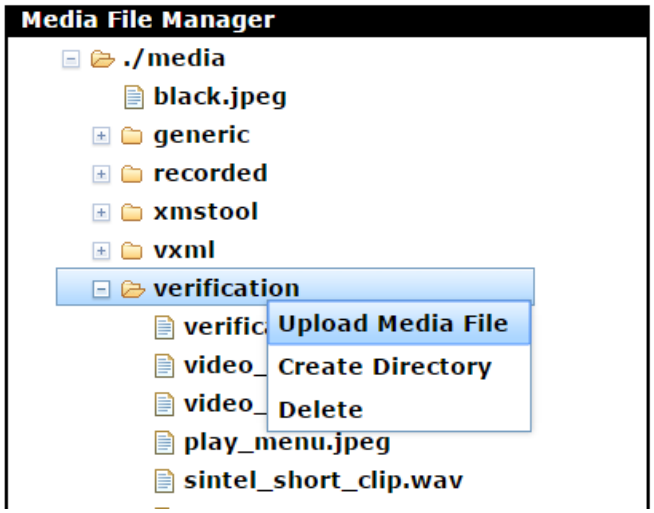
1. To install *Dialogic.mp4* manually, log in to PowerMedia XMS WebGUI, and then click **Media**.



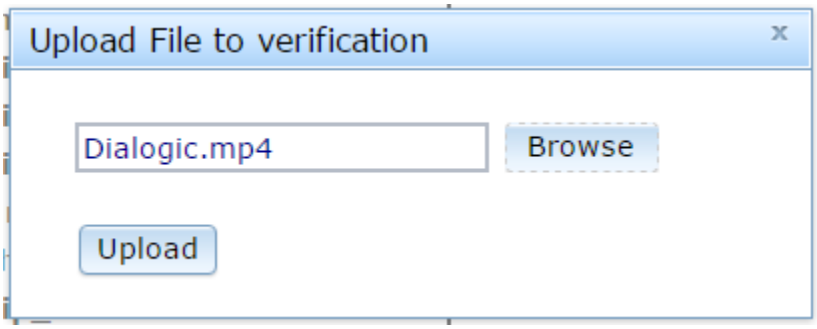
2. Click the **Media Management** tab.



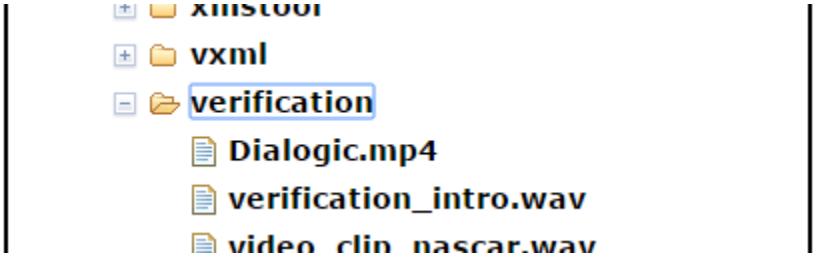
3. In the **Media File Manager** section, Right-click the **verification** folder and click **Upload Media File**.



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 Verification Application listens for incoming SIP and WebRTC calls and plays a *Dialogic.mp4* audio/video file. Based on the type of connection with the client (audio only, audio/video, or video only), it will play the appropriate media type to the client. For example, if the client supports an audio only connection, then only the audio portion of the sample .mp4 file will be played.

Follow these steps:

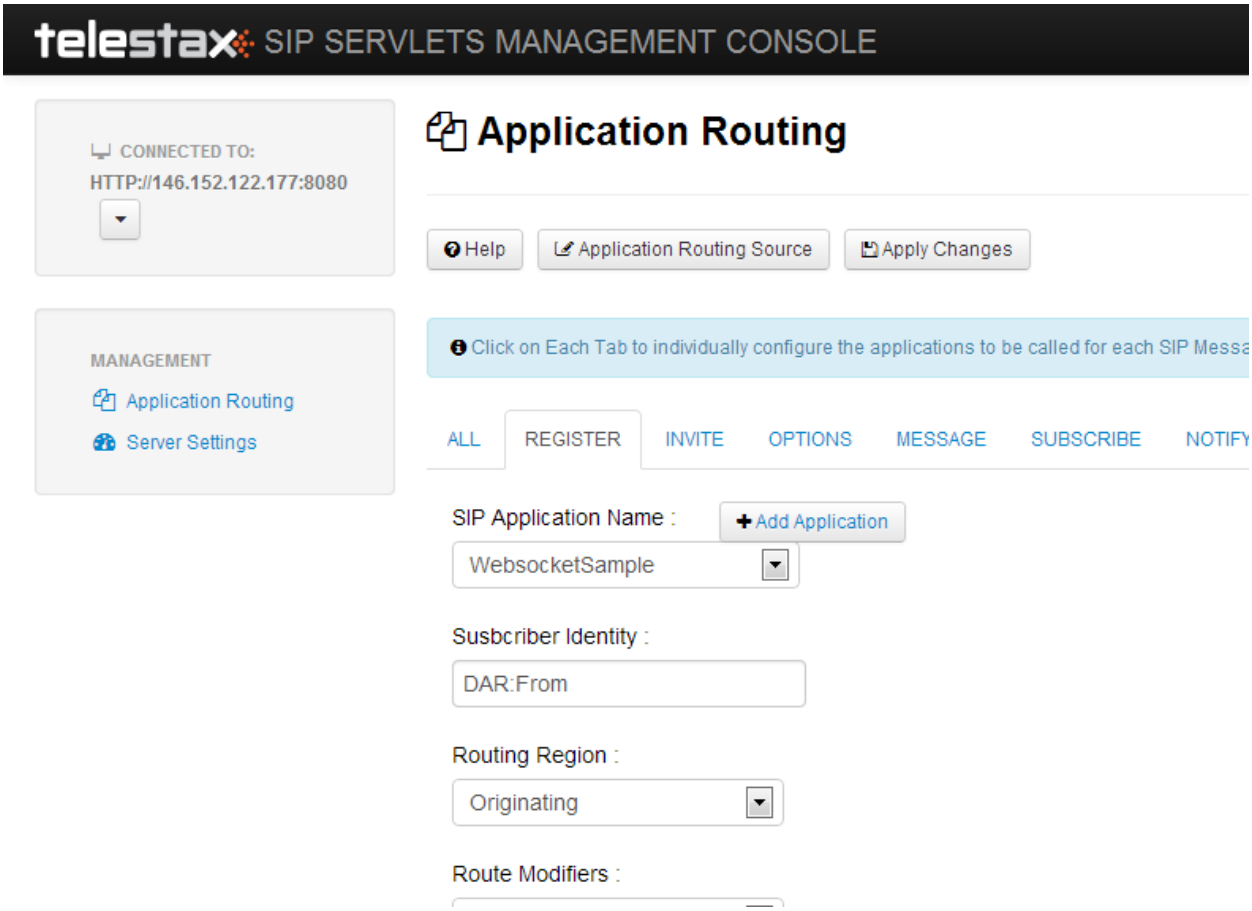
1. Configure Platform SIP Routing Using SIP Servlets Management
2. Dial in to the Dialogic JSR 309 Verification Demo

Configure Platform SIP Routing Using SIP Servlets Management

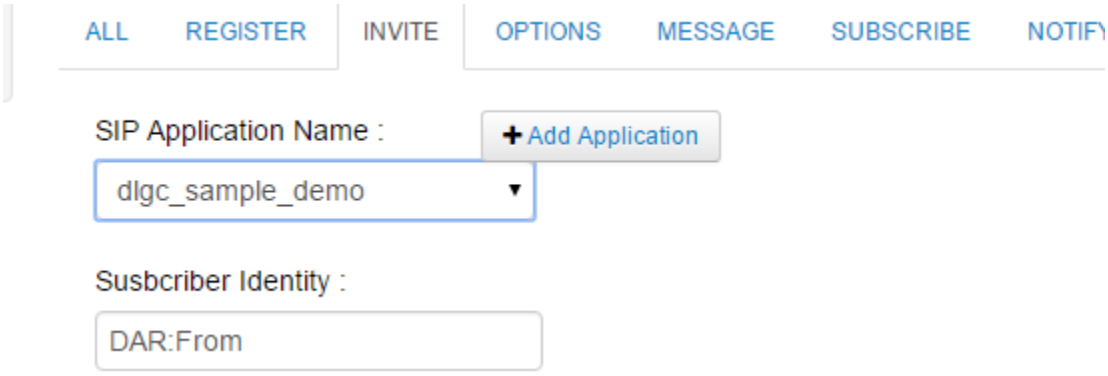
Before the deployed application can process SIP messages, it needs to be configured in the SIP Servlets Management console:

```
http://<AS_IP_Address>:8080/sip-servlets-management.
```

The following **Sip Servlets Management Console** appears.



Click **INVITE** and select **dlgc_sample_demo** in the **SIP Application Name** field.



Click **Apply Changes.** A message on the bottom of your screen will confirm its success: "[INFO] DAR Information successfully updated". The Application Server is configured to use the newly deployed application.

Dial in to the Dialogic JSR 309 Verification Demo

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

Dial in to a SIP client as follows:

1. Have a SIP client configured for the supported audio codec.
2. Place a call into the Application Server with the following URI:

   ```
   player@<as_ip_address>:5080
   ```

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

Dial in to a WebRTC client as follows:

Note: Standard based server side WebSockets are not supported in the JBoss platform. However, support was introduced for server side WebSockets in the next version of JBoss (renamed "WildFly"). TeleStax WildFly version is currently in Beta release.

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:
   https://<AS_Platform_IP>:8443/dlgc_sample_demo/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 a 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 the Dialogic JSR 309 Verification Application WAR file. The WAR package contains several necessary items. Refer to *build.xml* to get 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 *jboss-deployment-structure.xml* file used to exclude some automatic dependencies.
    - The *sip.xml* file, which defines SIP servlet-mapping for the deployment application WAR.

- The */DemoGUI* directory contains content for the application that supports WebRTC. It includes necessary .html and .js files for the verification demo. Note that this directory will not be present in application WAR files for JBoss because it does not support WebSockets.

Application Initialization Steps

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

```
package play;


@javax.servlet.sip.annotation.SipServlet(description = "Dialogic Sample Demo", applicationName =
"DialogicSampleDemo", name="AsyncPlayer", loadOnStartup=2)


@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:

1. Note the mandatory *@javax.servlet.sip.annotation.SipServlet* annotation. This annotation allows for the Sip Servlet metadata to be declared without having to create the deployment descriptor:
    a. description – Any string describing the application
    b. applicationName – Must match servlet mapping as defined in *sip.xml* for this application. The Dialogic JSR 309 Connector Verification Application *sip.xml* is shown below for reference:

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


  <app-name>DialogicSampleDemo</app-name>
  <display-name>DialogicSampleDemo</display-name>


  <session-config>
    <session-timeout>0</session-timeout>
  </session-config>


  <servlet-selection>
    <servlet-mapping>
      <servlet-name>AsyncPlayer</servlet-name>
```

```
      <pattern>
        <and>
          <equal>
            <var>request.method</var>
            <value>INVITE</value>
          </equal>
          <equal>
            <var>request.to.uri.user</var>
            <value>player</value>
          </equal>
        </and>
      </pattern>
    </servlet-mapping>
  </servlet-selection>


</sip-app>
```

c. name – The name of the SIP servlet class:

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

d. loadOnStartup – Defines the order in which the SIP container should start this SIP Servlet class. Since this application depends on JSR 309, it should start after the Dialogic JSR 309 Connector. Since Dialogic JSR 309 Connector Sip Servlet annotation is set (loadOnStartup) to 1, the application should be a number greater than 1.

2. 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.
3. 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:

```
protected boolean initDriver()
{
  dlgcDriver = DriverManager.getDriver(DLGC_309_DRIVER_NAME);
  PropertyInfo connectorProperty[] = dlgcDriver.getFactoryPropertyInfo();
…
```

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.
- 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.

The Verification Application configures the Dialogic JSR 309 Connector properties:

```
…
  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 Verification Application use the Apache Log4j2 (version 2) logging facility. The connector makes use of *log4j2.xml* file for its logging configuration. With the introduction of Log4j2, it is possible to change the log levels without stopping/restarting any components. All that needs to be done is open *log4j2.xml* and change the logging to the desired level. Log configuration file *log4j2.xml* can be found at:

```
${JBOSS_HOME}/standalone/configuration/Dialogic/log4j2.xml
```

As per *log4j2.xml* configuration, the Dialogic JSR 309 Connector and Verification Applications log output file *Dialogic.log* can be found in the *$JBOSS_HOME/logs/* directory.

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

```
<?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.

Dialogic JSR 309 Connector and Verification Application Troubleshooting

1. The Verification Application first opens the application properties. If the path is not set or the properties file does not exist, the DEBUG log file will show an error as follows:

```
13:53:55.851 INFO  play.AsyncPlayer 136 init - init() - Entering
13:53:55.853 DEBUG play.AsyncPlayer 138 init - init() - servletName: AsyncPlayer
13:53:55.854 INFO  base.ConfigProperty 40 <init> - ConfigProperty() - Entering
13:53:55.854 INFO  base.ConfigProperty 56 LoadProperties - LoadProperties() - Entering
13:53:55.855 INFO  base.ConfigProperty 60 LoadProperties - LoadProperties() - Properties
File = /home/jboss/mss-3.1.633-jboss-as-
7.2.0.Final/standalone/configuration/Dialogic/dlgc_sample_demo.properties
13:53:55.855 ERROR base.ConfigProperty 77 LoadProperties - java.io.FileNotFoundException:
/home/jboss/mss-3.1.633-jboss-as-
7.2.0.Final/standalone/configuration/Dialogic/dlgc_sample_demo.properties (No such file
or directory)
13:53:55.856 ERROR base.ConfigProperty 78 LoadProperties - LoadProperties() - base
Configuration File: /home/jboss/mss-3.1.633-jboss-as-
7.2.0.Final/standalone/configuration/Dialogic/dlgc_sample_demo.properties load failed
13:53:55.856 INFO  base.ConfigProperty 81 LoadProperties - LoadProperties() - Exiting
```

Successful loading of the properties file will be shown as an INFO message as follows:

```
13:55:46.315 INFO  play.AsyncPlayer 136 init - init() - Entering

13:55:46.317 DEBUG play.AsyncPlayer 138 init - init() - servletName: AsyncPlayer

13:55:46.318 INFO  base.ConfigProperty 40 <init> - ConfigProperty() - Entering

13:55:46.319 INFO  base.ConfigProperty 56 LoadProperties - LoadProperties() - Entering

13:55:46.320 INFO  base.ConfigProperty 60 LoadProperties - LoadProperties() - Properties
File = /home/jboss/mss-3.1.633-jboss-as-
7.2.0.Final/standalone/configuration/Dialogic/dlgc_sample_demo.properties

13:55:46.321 INFO  base.ConfigProperty 73 LoadProperties - LoadProperties() - base
Configuration File: /home/jboss/mss-3.1.633-jboss-as-
7.2.0.Final/standalone/configuration/Dialogic/dlgc_sample_demo.properties Successfully
Loaded

13:55:46.322 INFO  base.ConfigProperty 81 LoadProperties - LoadProperties() - Exiting
```

2. Make sure that the Dialogic Connector SIP Servlet gets initialized first. If successful, you will see following DEBUG print:

```
09:58:49.959 DEBUG com.vendor.dialogic.javax.media.mscontrol.spi.DlgcDriver 147
registerDialogic309Driver - DlgcDriver::registerDialogic309Driver() - Application
Platformloading..loading driver now

09:58:49.959 DEBUG com.vendor.dialogic.javax.media.mscontrol.spi.DlgcDriver 148
registerDialogic309Driver - DlgcDriver:registerDialogic309Drive() calling
DriverManager.re

09:58:49.961 DEBUG com.vendor.dialogic.javax.media.mscontrol.spi.DlgcDriver 150
registerDialogic309Driver - DlgcDriver:registerDialogic309Drive() returned from
DriverMana

09:58:49.962 DEBUG com.vendor.dialogic.javax.media.mscontrol.sip.DlgcSipServlet 137 init
- Dialogic Servlet Initialized...
```

The Verification Application will take the new set of required parameters and create a the Dialogic JSR 309 Factory. Successful creation of the JSR 309factory will be shown in the DEBUG logs as follows:

```
13:55:46.700 DEBUG com.vendor.dialogic.javax.media.mscontrol.spi.DlgcDriver 408
getControlFactory - DlgcDriver::getControlFactory() property passed in:

13:55:46.706 DEBUG com.vendor.dialogic.javax.media.mscontrol.DlgcMsControlFactory 103
<init> - DlgcMsControlFactory:: CTOR using Dynamic Factory Configuration

13:55:46.709 DEBUG com.vendor.dialogic.javax.media.mscontrol.sip.DlgcMediaServer 235
<init> - DlgcMediaServer CTOR supporting Dynamic Configuration:

13:55:46.710 DEBUG com.vendor.dialogic.javax.media.mscontrol.sip.DlgcMediaServer 251
<init> - DlgcMediaServer CTOR using the following XMS SIP Values:  username: msml= Media
Server IP: 146.152.122.4 Media Server SIP Port: 5060

13:55:46.711 DEBUG com.vendor.dialogic.javax.media.mscontrol.sip.DlgcMediaServer 252
<init> - DlgcMediaServer CTOR using the following XMS Connector AS SIP Values: AS Server
IP: 146.152.122.146 Connector AS SIP Port: 5080
```

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.

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 (Version used here: Mars.1 Release (4.5.1)).
- In order to build provided demo applications, obtain two TeleStax platform dependent libraries that NOT provided as part of the Dialogic JSR 309 Connector distribution package. They can be found under the following directories:

  {JBOSS_HOME}/modules/system/layers/base/org/mobicents/javax/servlet/sip/main

  {JBOSS_HOME}/modules/system/layers/base/javax/servlet/api/main

  - (TelScale)
    - *sip-servlets-spec-7.0.2.GA-TelScale.jar*
    - *jboss-servlet-api_3.0_spec-1.0.2.Final.jar*
  - (Mobicents)
    - *sip-servlets-spec-3.0.xxx.jar (xxx represents version of Mobicents 633 for example)*
    - *jboss-servlet-api_3.0_spec-1.0.2.Final.jar*

Creating the Build Environment

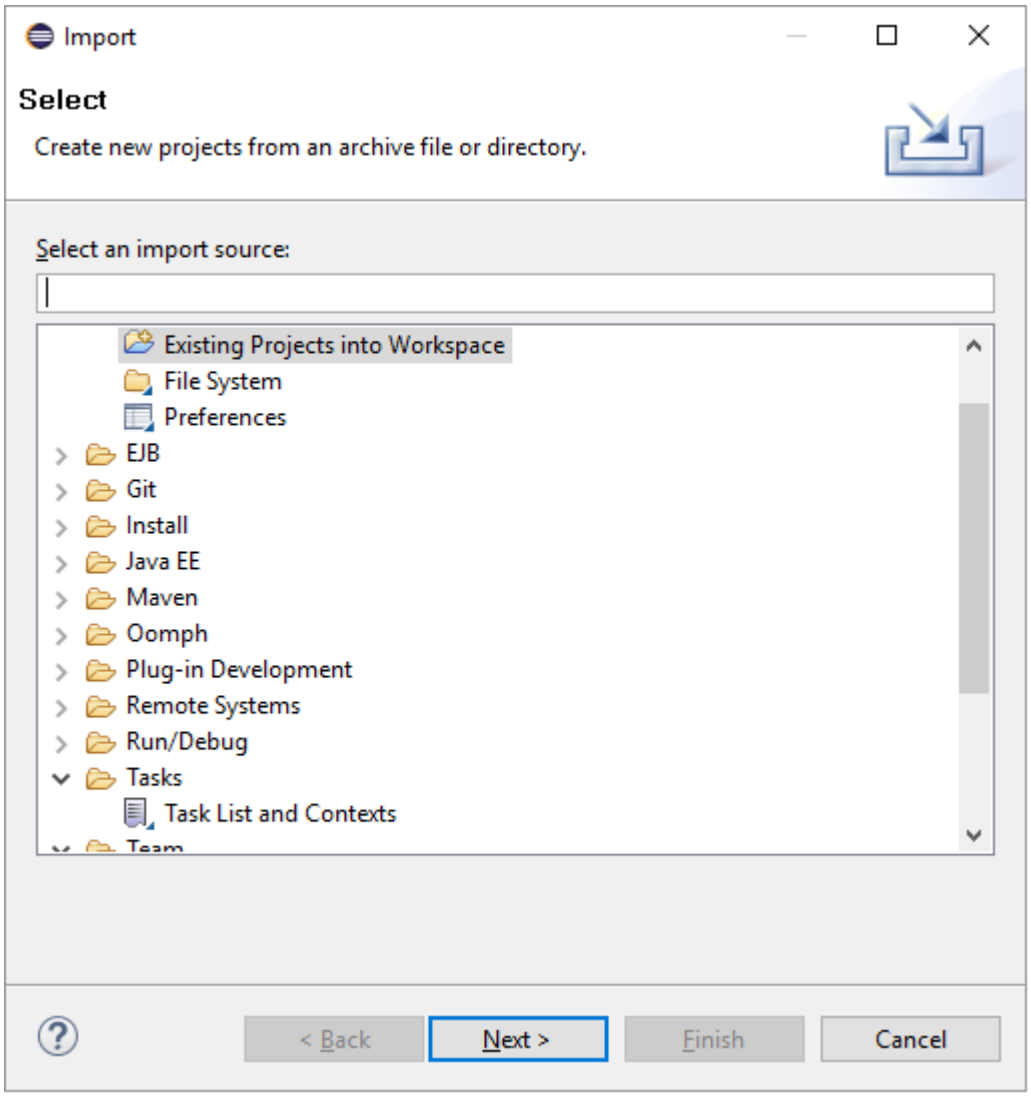
The Dialogic JSR 309 Verification Application source project comes with all the necessary components to compile and build the application WAR file. Follow these steps to create a Dialogic demo build environment:

Prepare the Eclipse Workspace

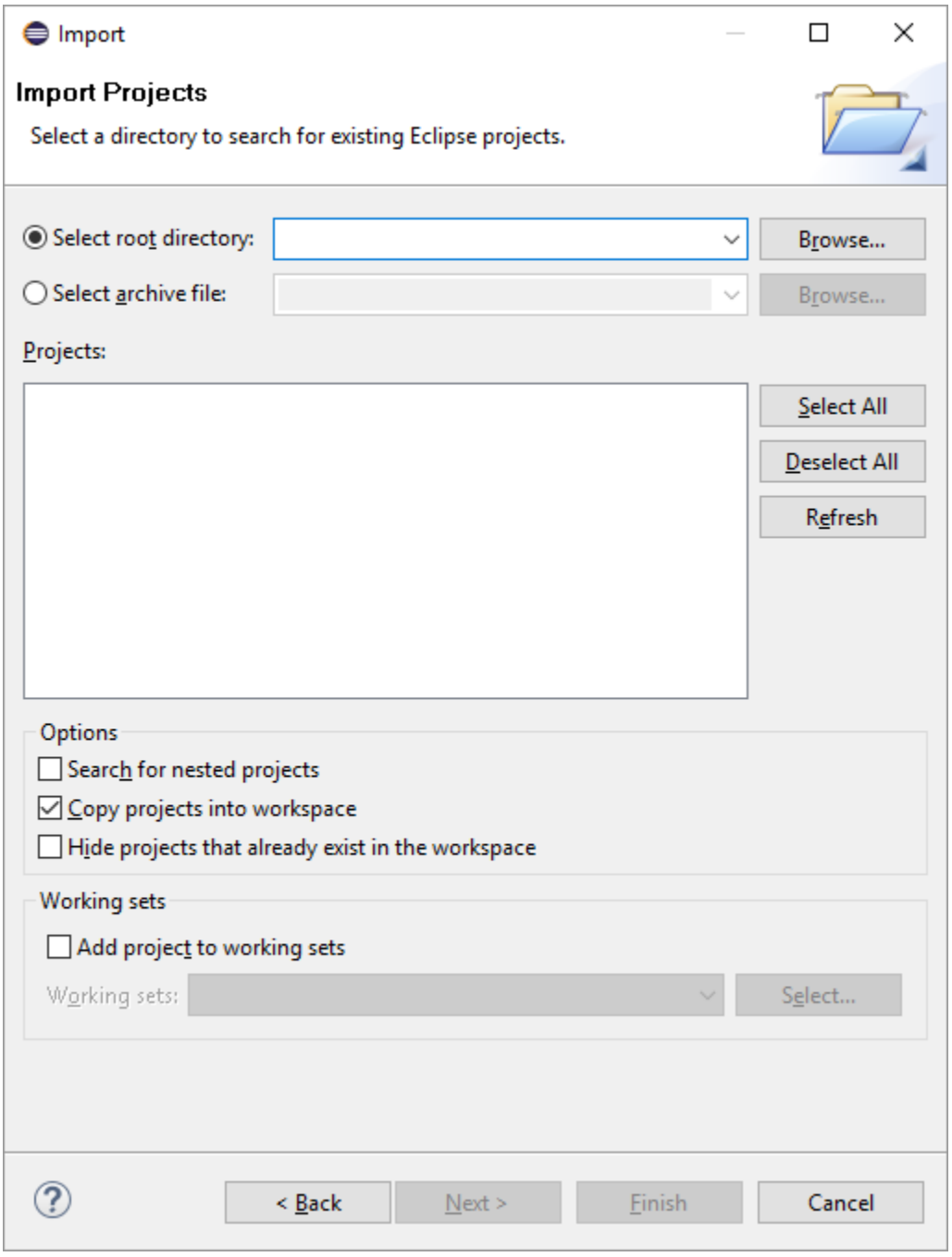
First, prepare the Eclipse workspace:

1. From the distribution package, copy the *DlgcJSR309/application/Project* directory and its content to a known location on your system.
2. Verify that the *Project/lib/3rdParty* folder contains all required third-party JAR files including the Dialogic JSR 309 Connector JAR files.
3. Copy the required Application Server platform specific libraries into the *Project/lib/AS* directory.
4. Open Eclipse IDE and click **File > Import**.

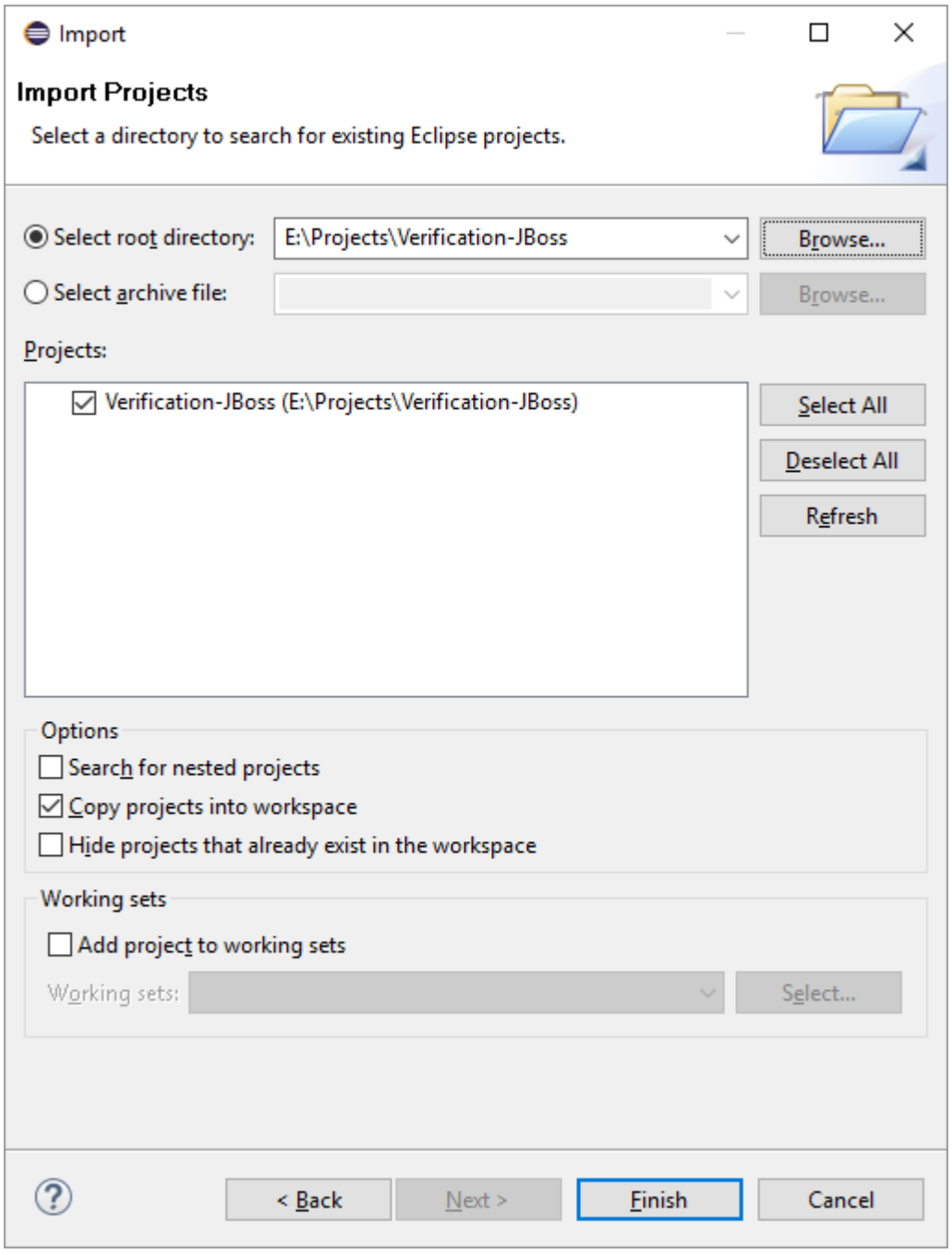
5. Select **Existing Project into Workspace**, and then click **Next**.



6. Click **Browse** and navigate to the *Project* directory on the system.



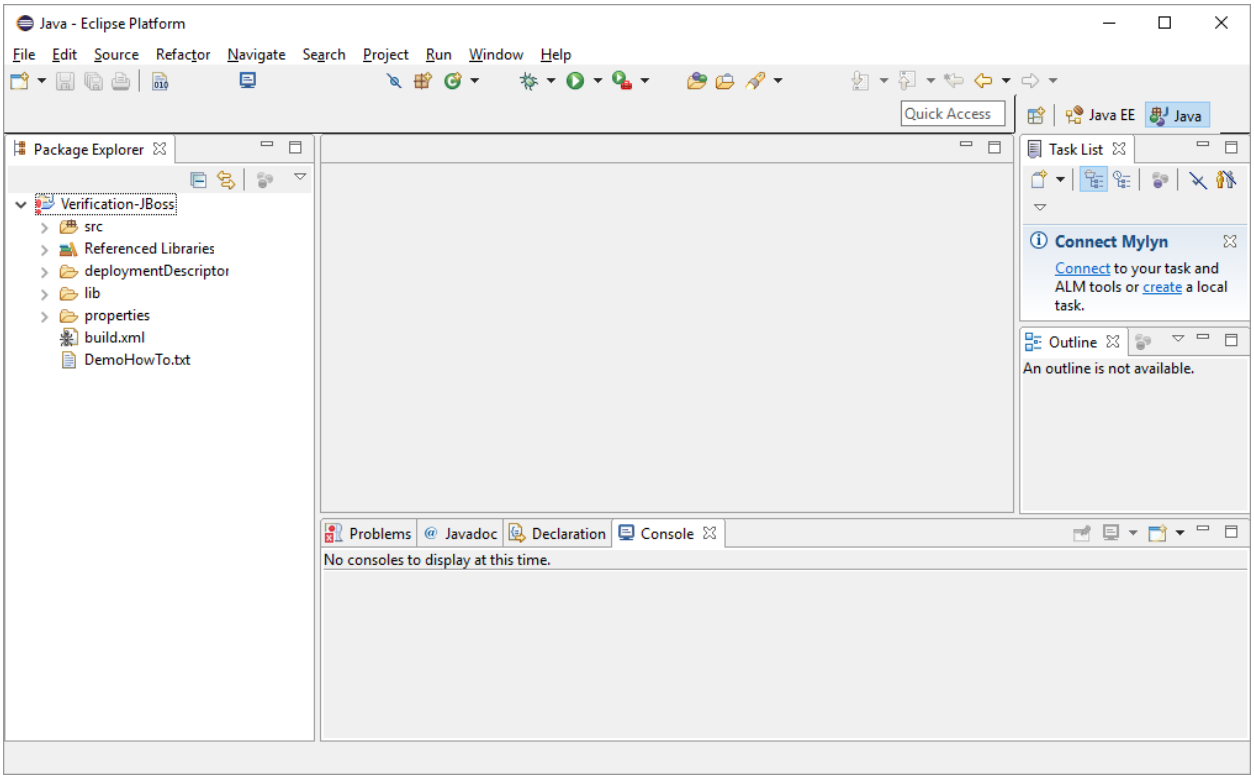
7. Click **Finish**. The Project has been imported to the Eclipse workspace.



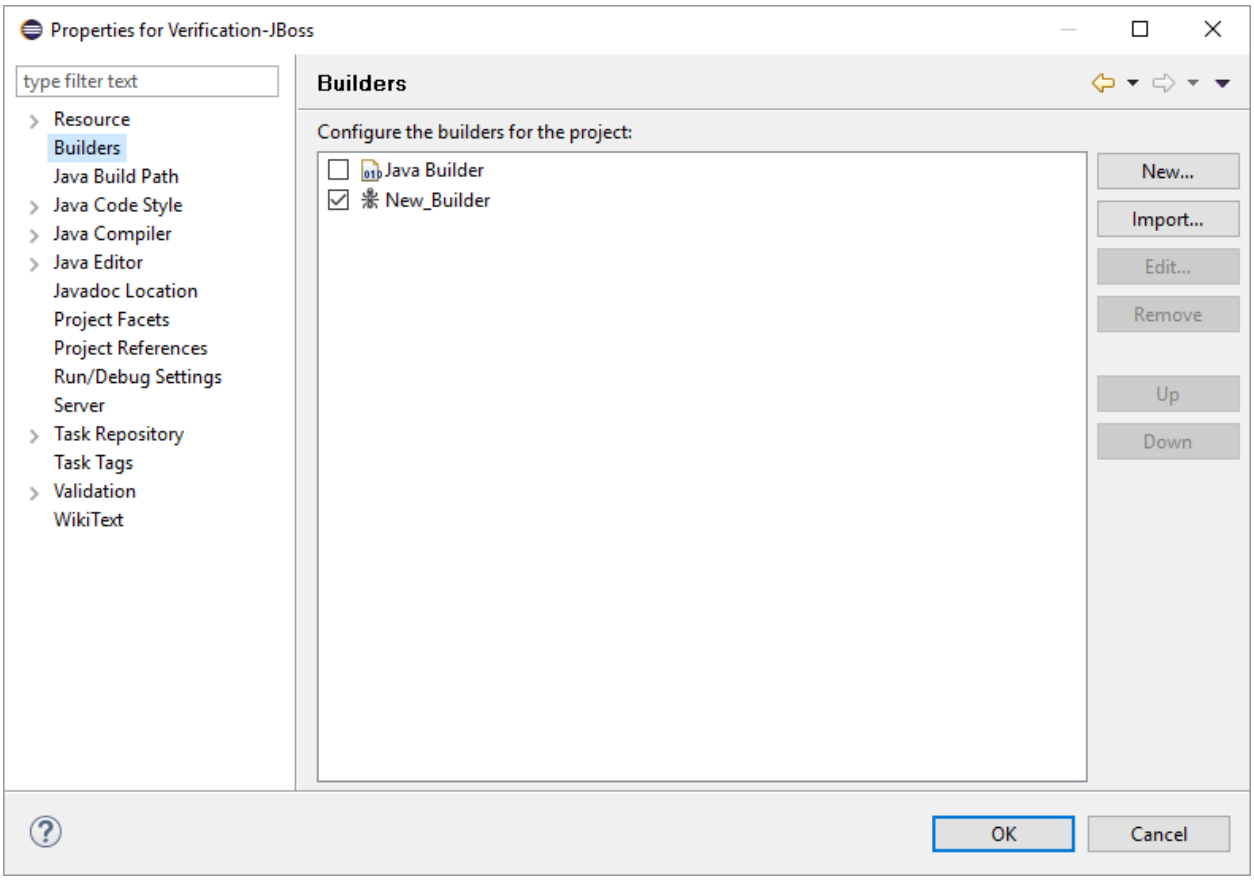
Configure the Application

To configure the application, edit the following settings:

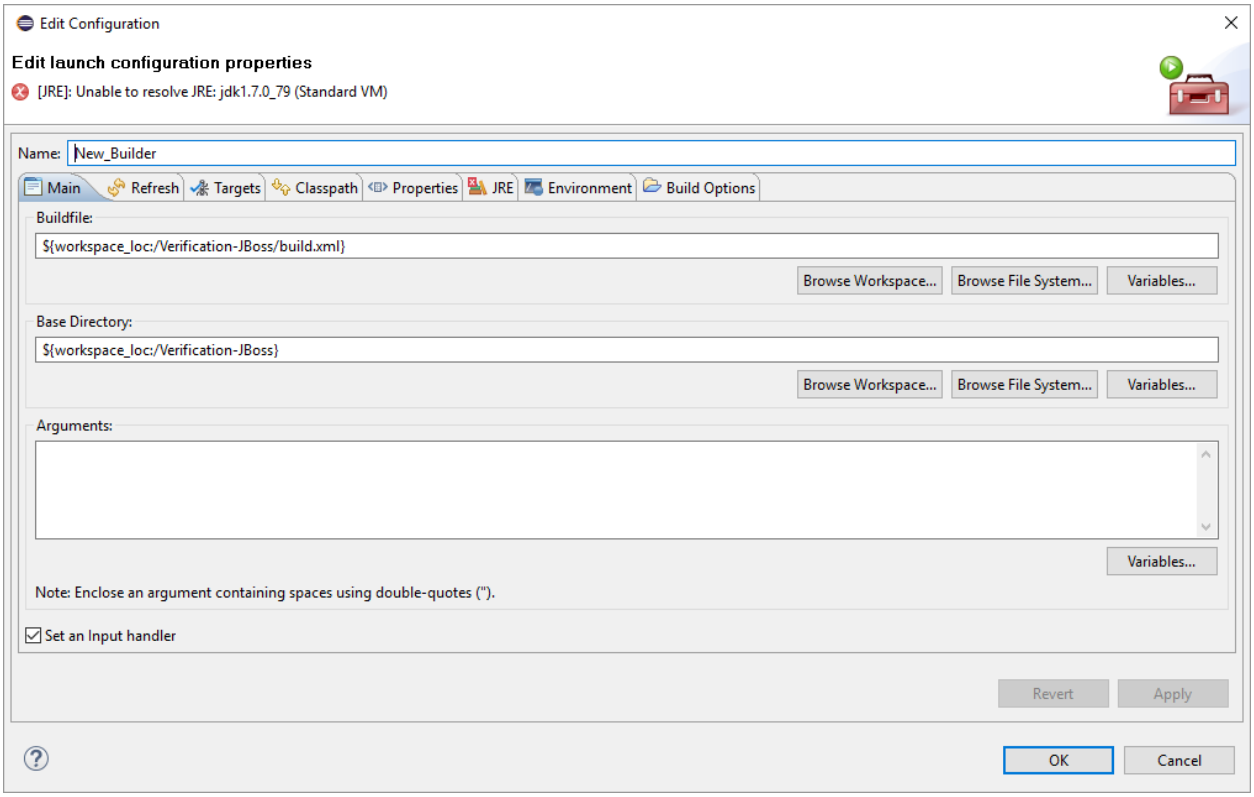
1. Right-click the **Verification-JBoss** project folder (for this example) and select **Properties**.



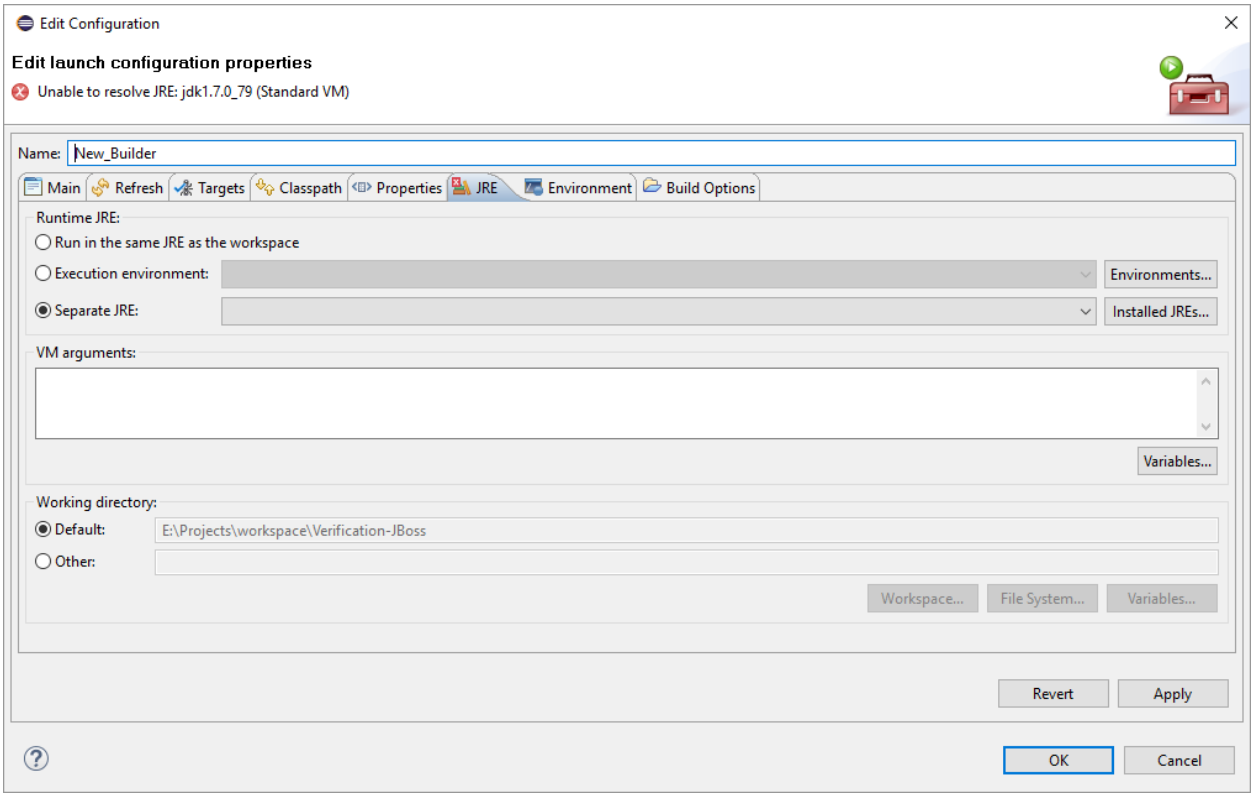
2. Click **Builders** and then double-click **New_Builder**.



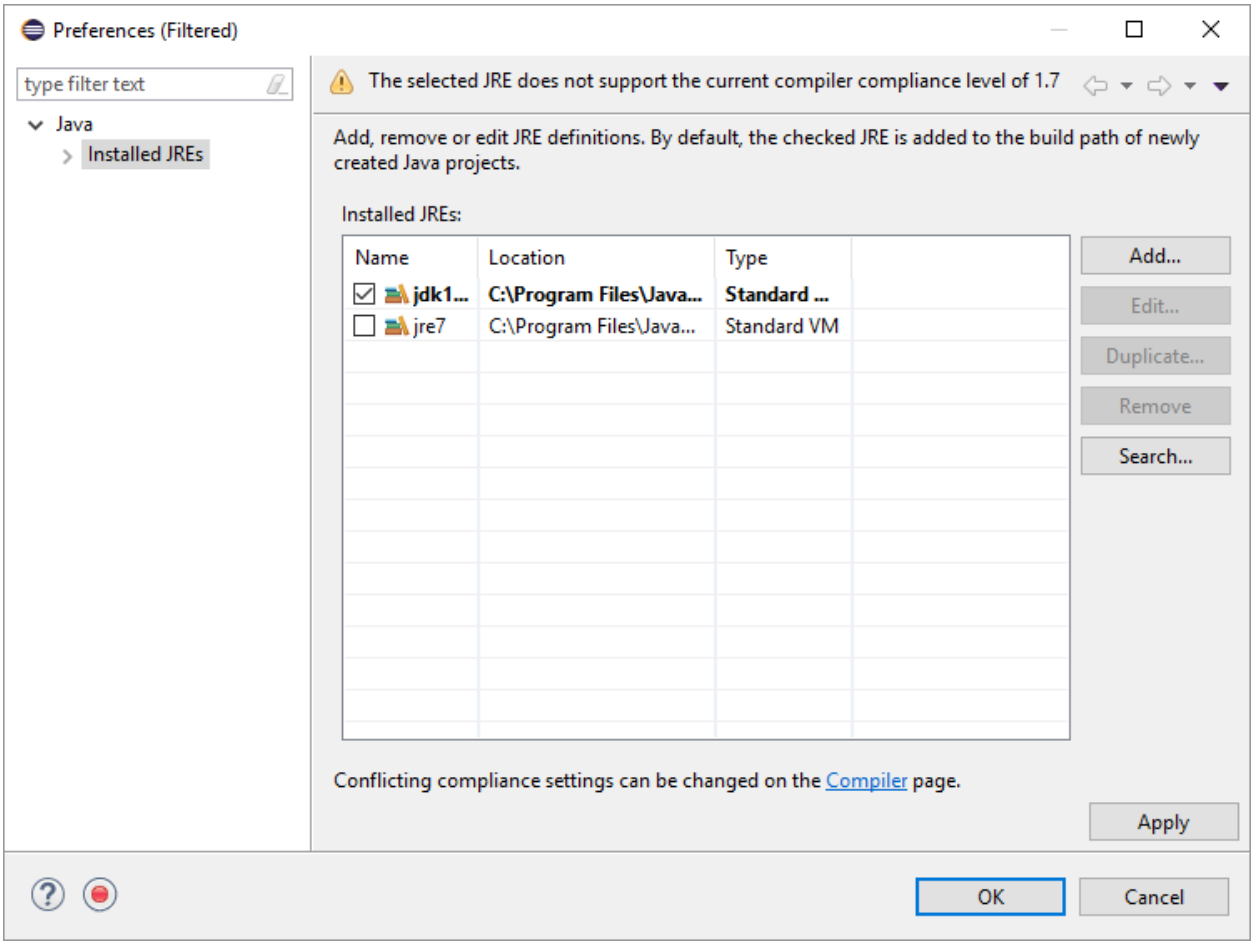
3. Verify that **Buildfile** and **Base Directory** point to correct workspace *build.xml* and directory.



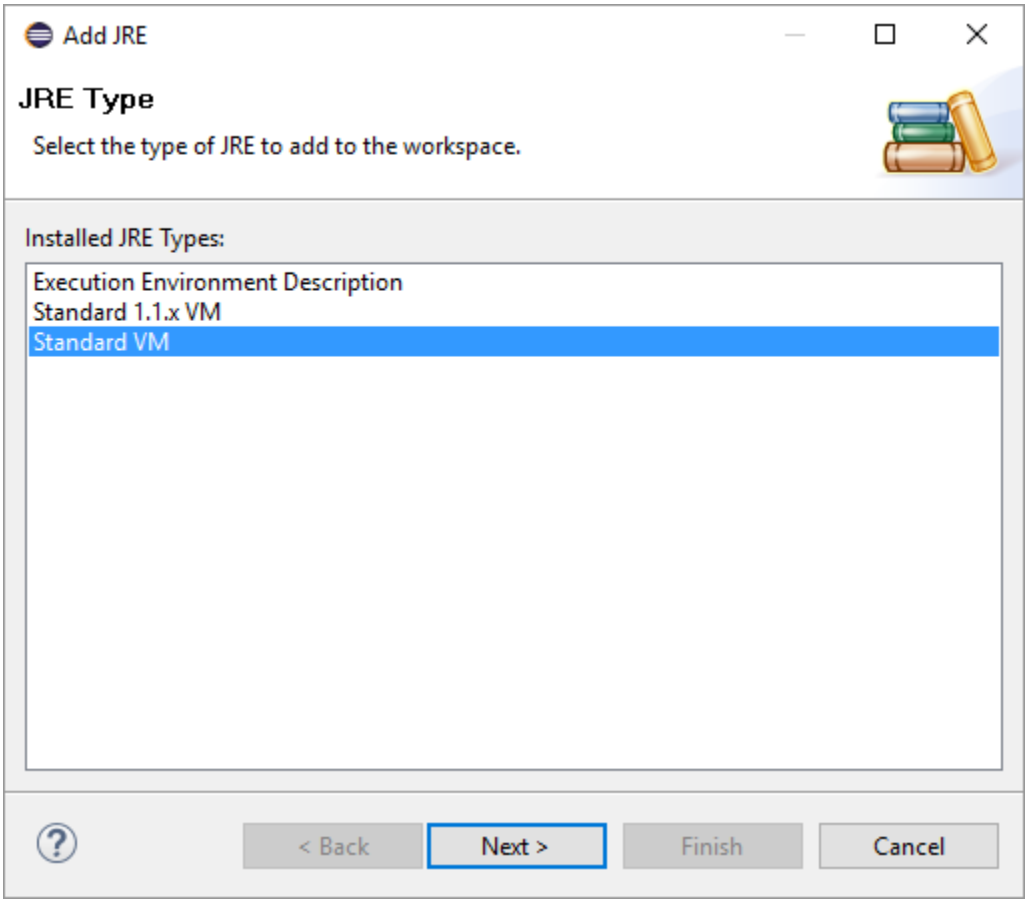
4. Click the **JRE** tab.



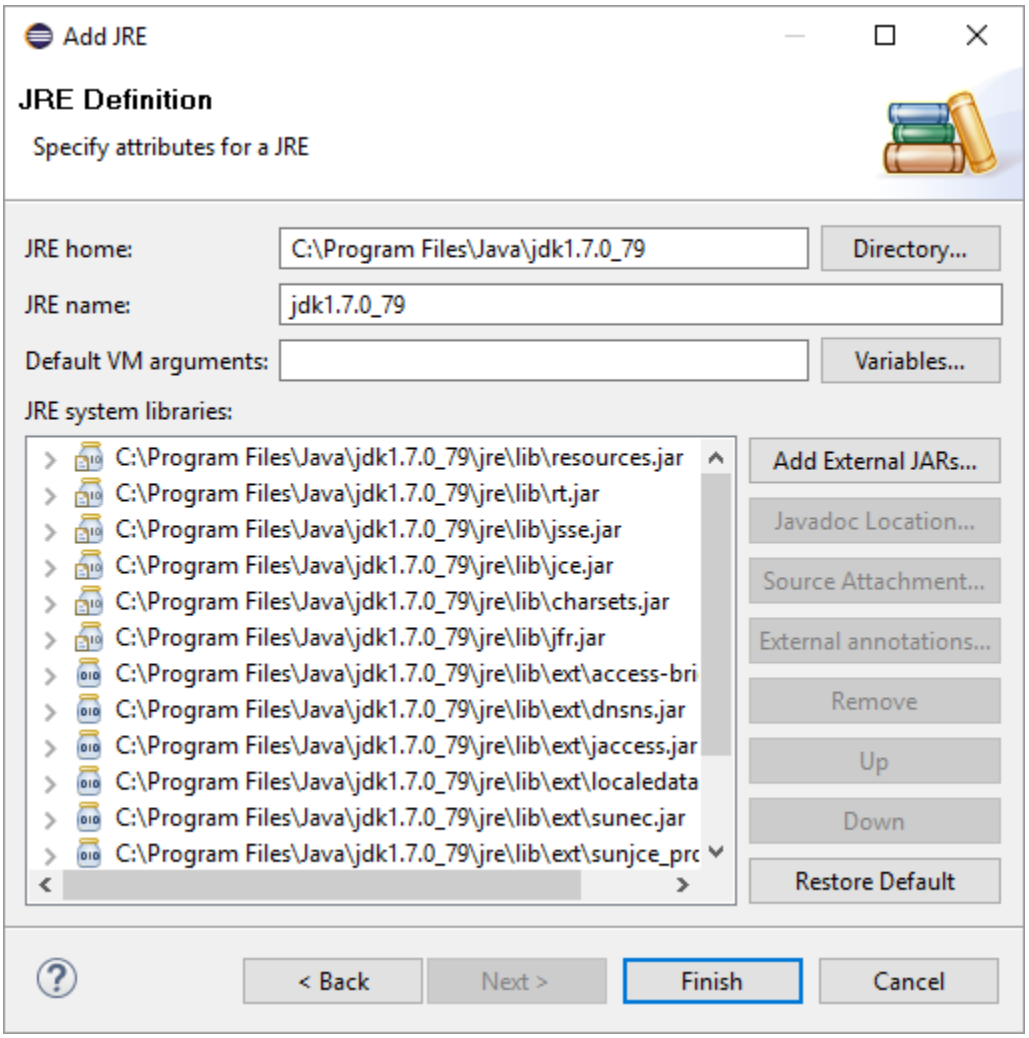
5. In the **Separate JRE** drop-down menu, select the appropriate JDK as required by the platform. If not present there, click **Installed JREs** and then click **Add**.



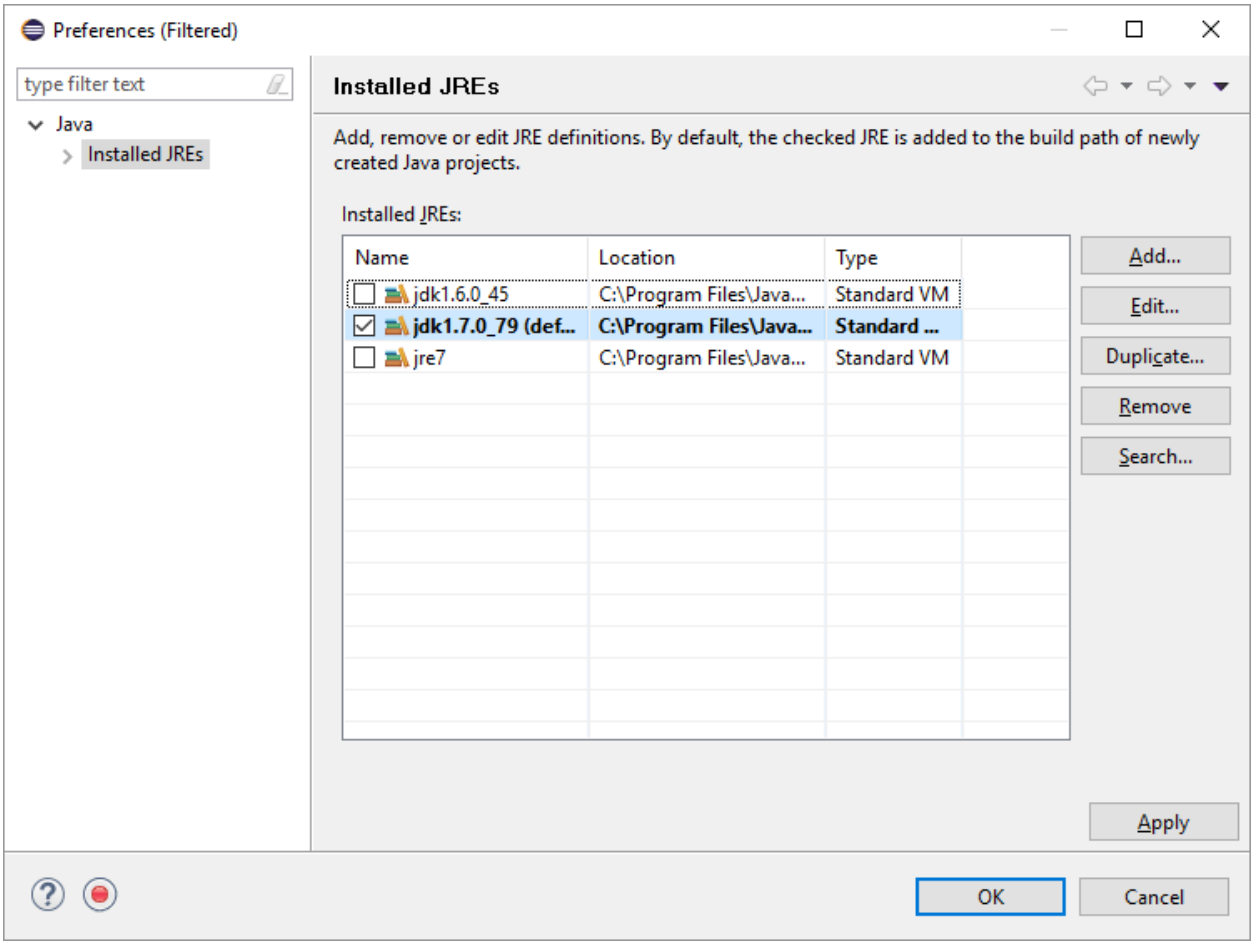
6. Click **Next**.


Add JRE
JRE Type
Select the type of JRE to add to the workspace.
Installed JRE Types:
Execution Environment Description
Standard 1.1.x VM
Standard VM
< Back
Next >
Finish
Cancel

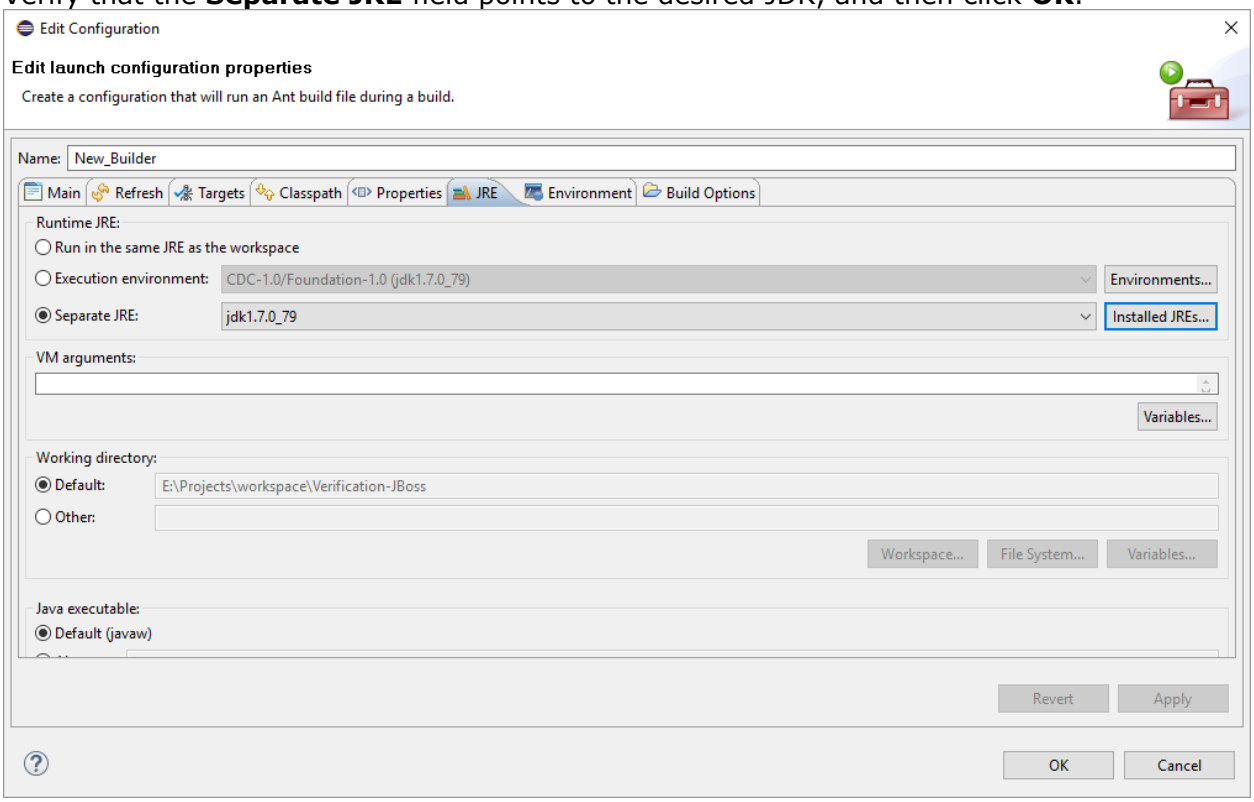
7. Click **Directory** and navigate to the appropriate JDK. When done, click **Finish**.



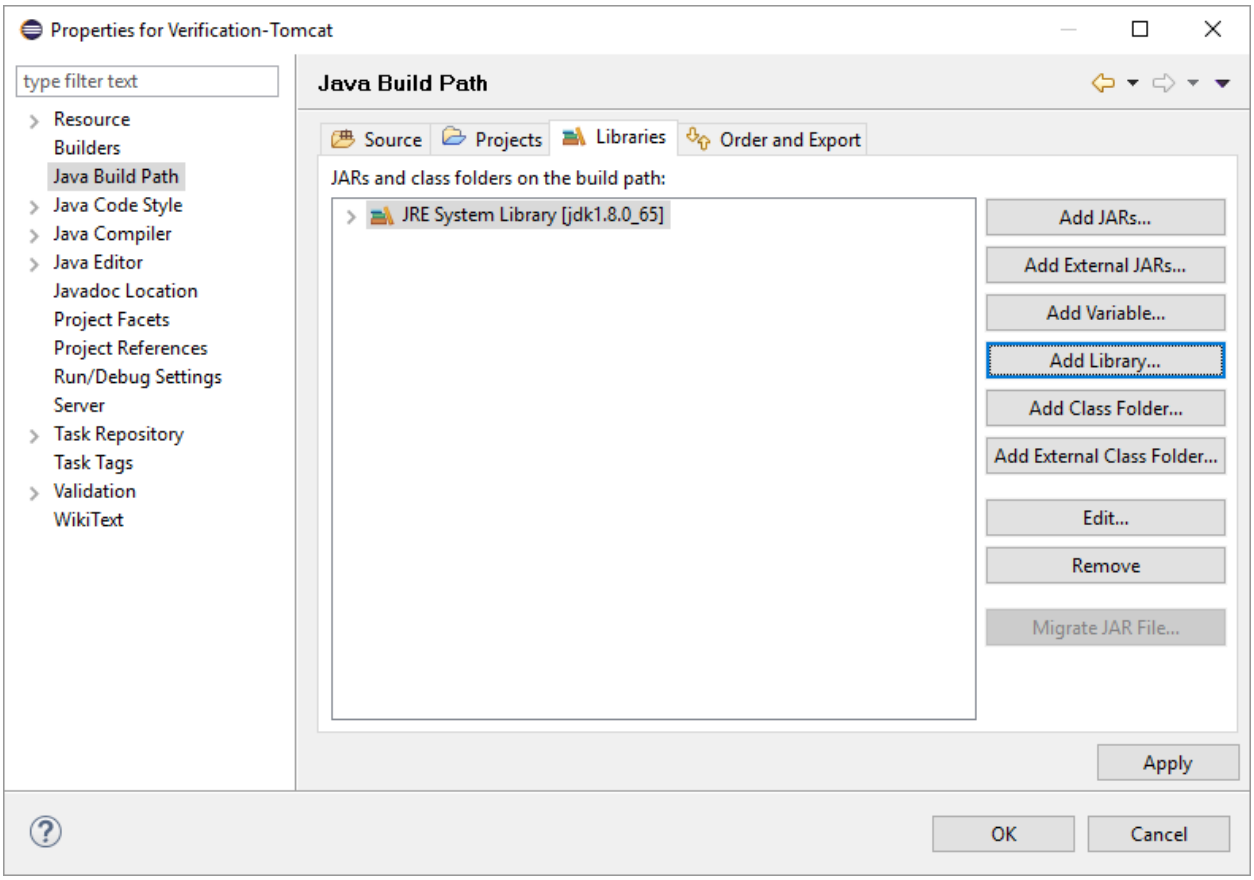
8. Make sure that appropriate JDK is chosen, and then click **OK**. In this example, jdk1.7.0_79 is chosen.



9. Verify that the **Separate JRE** field points to the desired JDK, and then click **OK**.

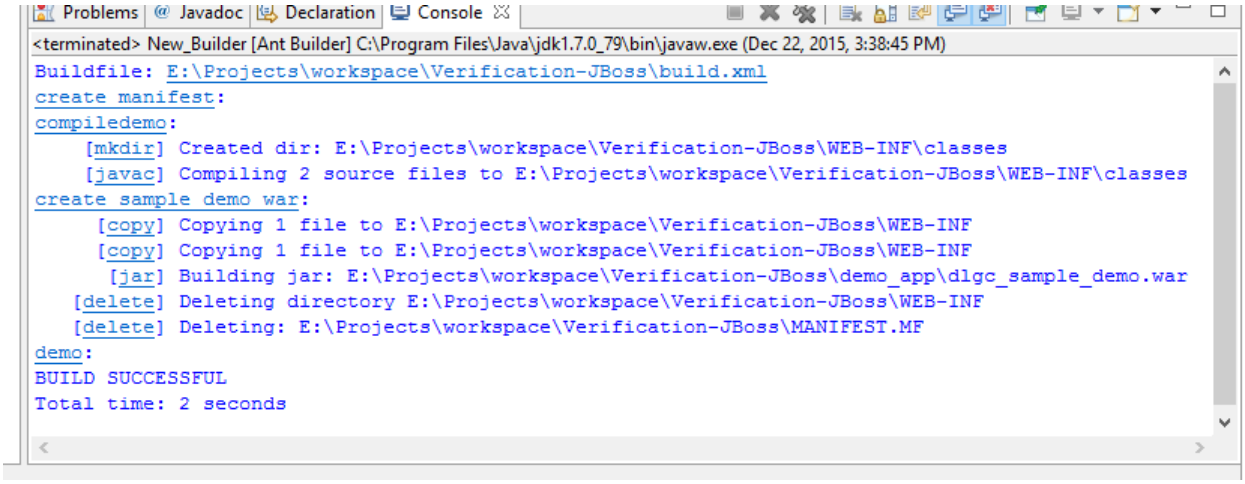


10. Click **Java Build Path** and verify that all JAR files are correctly pointed to. If not, click ***Add JARs*** and select all the JAR files from within *Project/lib* folders. Once that is done, click **OK**.



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\demo_app* directory named *dlgc_sample_demo.war*. In order to deploy this application, follow the same deployment instructions as described in Installation and Configuration of the Dialogic JSR 309 Connector Demo.

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

In order to connect the newly created project to the deployed WAR file in the Application Server debugging purposes, developers need to do the following:

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

Configuring the Application Server Platform for Remote Debugging

Configure the Application Server platform for remote debugging as follows:

1. Stop the Application Server.
2. Edit *standalone.conf:*

```
${JBOSS_HOME}/bin/standalone.conf
```

3. Find the line below and uncomment it as illustrated below:

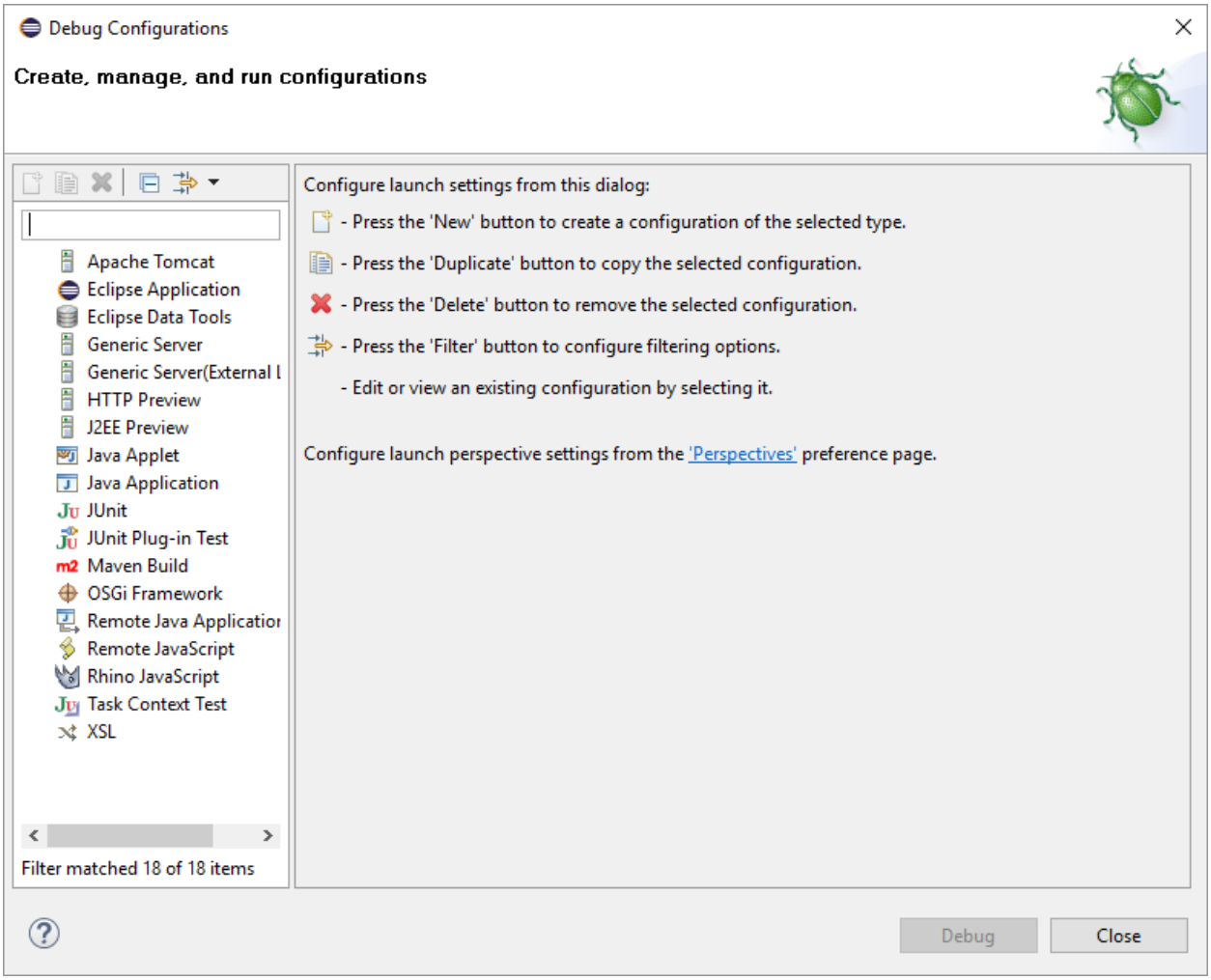
```
# Sample JPDA settings for remote socket debugging
JAVA_OPTS="$JAVA_OPTS -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n"
```

Note: The socket address specified above is 8787 by default 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.

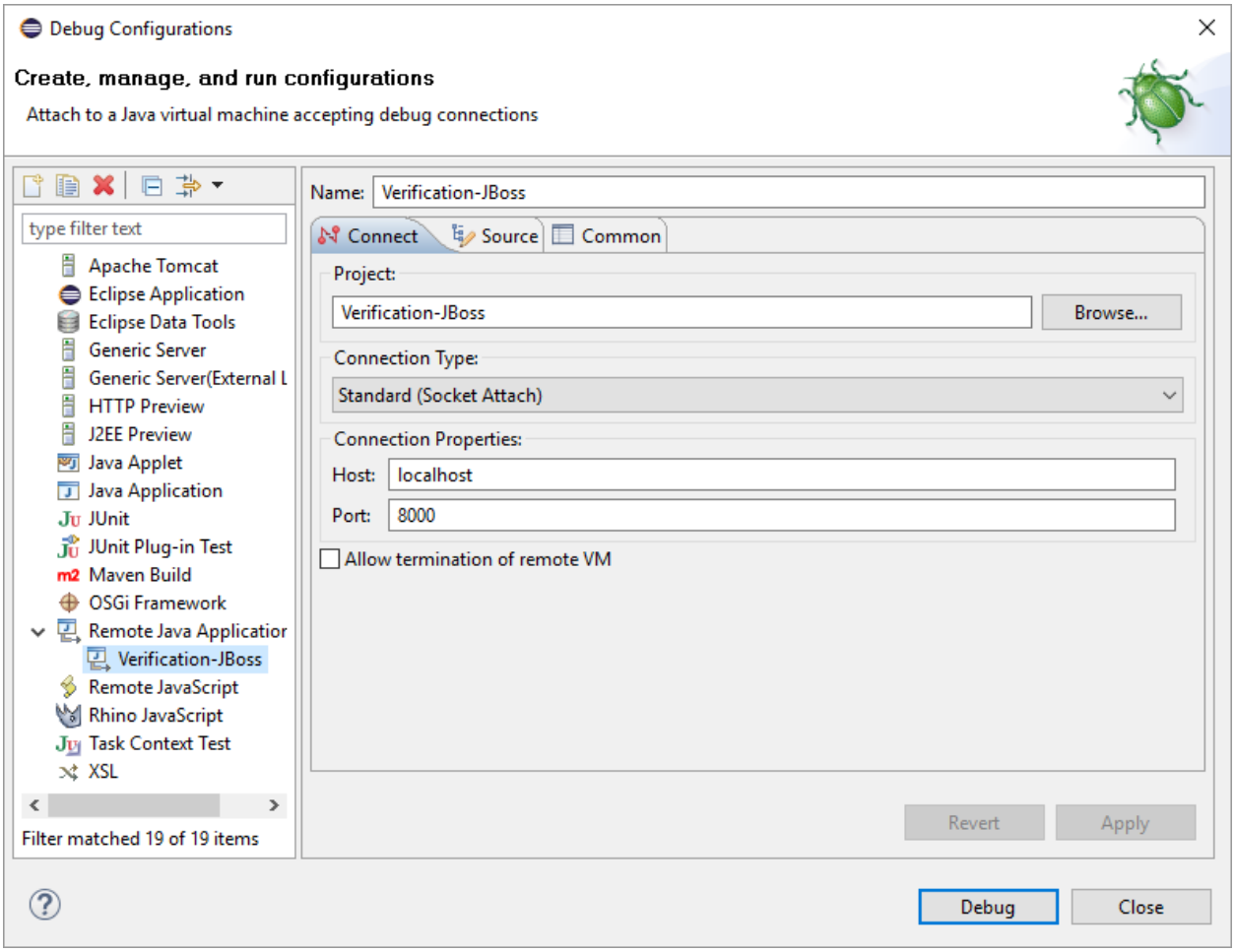
4. Start the Application Server and make sure that there are no errors in the console.

Eclipse Project Configuration for Remote Debugging

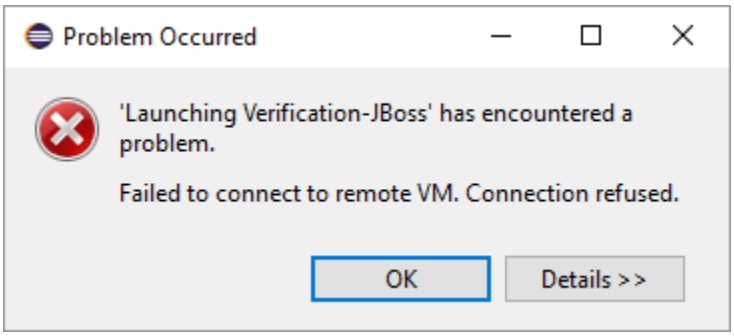
To configure the existing and working Dialogic JSR 309 Connector project, the remote debugging section needs to be configured. In Eclipse, go to the **Run** menu and click **Debug Configurations**.



1. In the **Debug Configurations** window, double-click **Remote Java Application**. The project name should be automatically added.

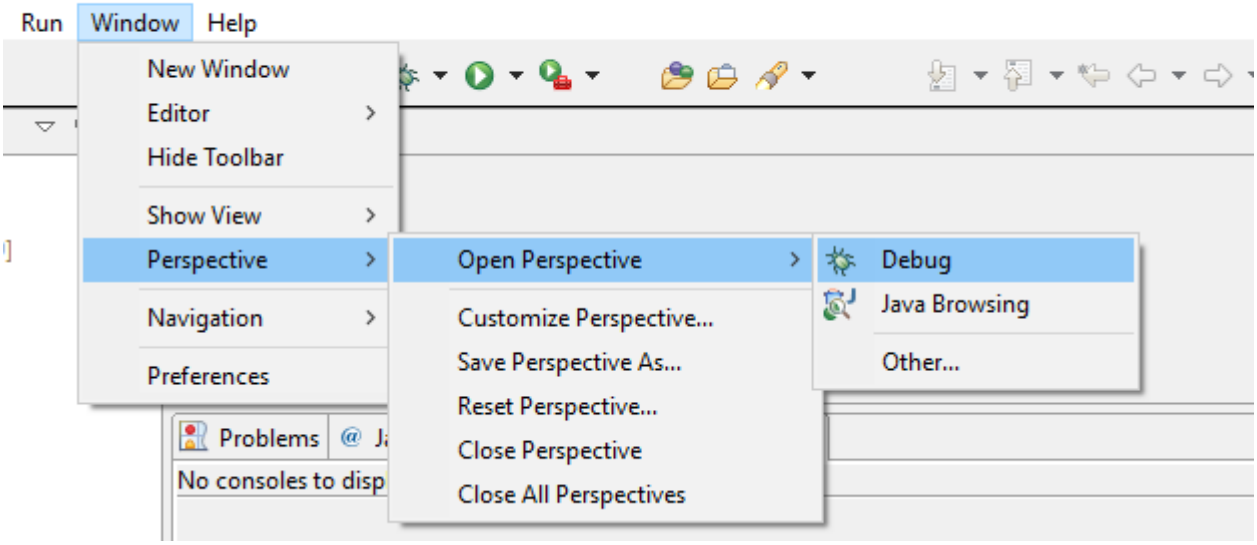


2. In the **Connection Properties** field, specify the host address and the port address of the J2EE platform with the deployed application for the debugger to connect to.
3. Click **Debug**. Provided that Application Server is already running and that debugging port 8787 is accessible, remote debugging should connect. If not successful, the following message appears. Check your configuration and try again.

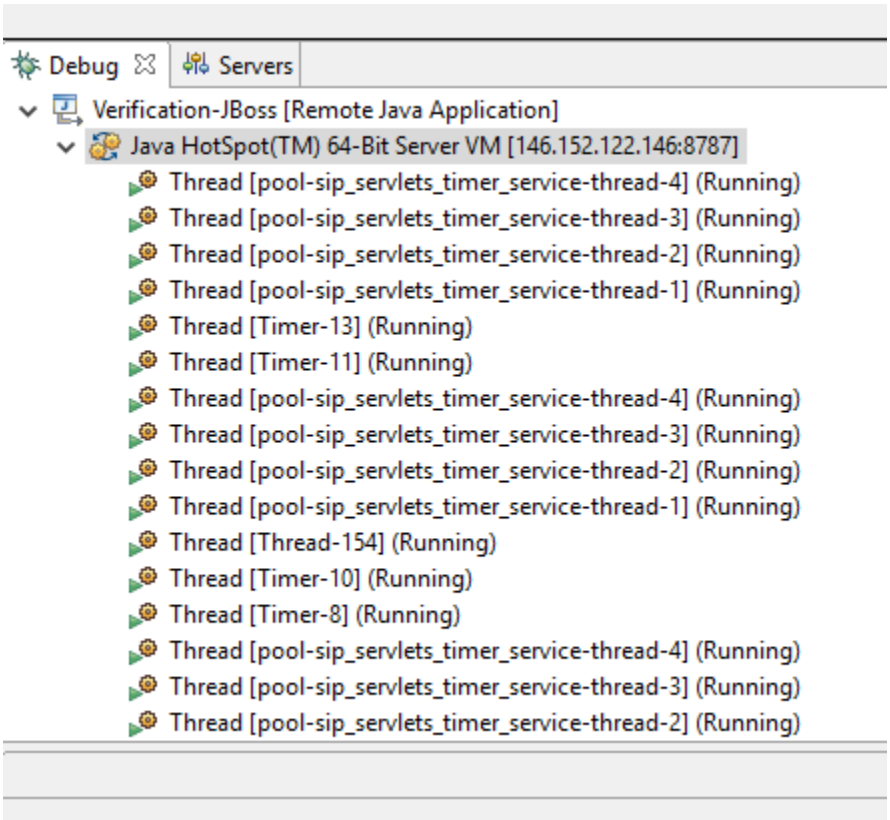


Note: If you are already connected and you try to connect again, the same error will be reported.

The way to verify that debugging session has been established between Eclipse and the Application Server platform is to click **Window > Perspective > Open Perspective > Debug**.



Once on the **Debug** screen, a successful debugging connection will be shown as follows. The Eclipse project is connected to the build application that is deployed in the TeleStax Application Server**.**



7. Appendix A: Dialogic JSR 309 Connector Environment Setup

This section describes, in detail, how to set up the Dialogic JSR 309 Connector environment.

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

This section does not go into the details of the Application Server platform, but it will help build the Application Server quickly for use.

It should be noted that OS level configuration should include the following:

- Enable NTP (Network Time Protocol)
- Enable ports in firewall (if applicable)

  Note: The following IP ports must be enabled in the firewall for the system to operate correctly: 8080 (TCP), 9990 (TCP), 5080 (UDP and TCP), and optional remote debugging port 8787 (TCP).

If you need further details on TeleStax Application Server, visit www.telestax.com.

Installing and Configuring the TeleStax JBoss Application Server

Note: If you are familiar with TeleStax AS or are planning to deploy on an existing TeleStax setup, proceed to Installing the Dialogic JSR 309 Connector.

This section describes the installation and configuration instructions for the Application Server:

- Preinstallation Setup
- TeleStax Installation
- TeleStax Configuration
- Firewall Configuration
- TeleStax Startup
- TeleStax Verification

Preinstallation Setup

Install the OS supported by TeleStax. Refer to www.telestax.com for details. For the purpose of this documentation, a CentOS 6.4/6.5 64-bit operating system with minimum installation options was used. Follow the steps below:

1. Log in to the newly installed operating system and install zip/unzip package:

   ```
   yum install zip unzip
   ```

2. Copy and install the latest 1.7 version of JDK .rpm package, which can be downloaded from www.oracle.com.

   ```
   rpm –ivh jdk-7u80-linux-x64.rpm
   ```

3. Under the user home root directory, edit the .bashrc file and include the following export lines:

   For TelScale AS:

   ```
   export JAVA_HOME=/usr/java/jdk1.7.0_80
   export JBOSS_HOME=/home/jboss/TelScale-SIP-Servlets-7.0.3.329-jboss-as-7.2.0.Final
   ```

   For Mobicents AS:

   ```
   export JAVA_HOME=/usr/java/jdk1.7.0_80
   export JBOSS_HOME=/home/jboss/mss-3.1.633-jboss-as-7.2.0.Final
   ```

4. Save the file and execute the following command for the changes to take effect:

   ```
   source /home/jboss/.bashrc
   ```

5. Edit the */etc/hosts* file and add a line at the very top of the file that corresponds to your system's IP address and the hostname.

   Here is an example of *hosts* file content:

   ```
   xxx.xxx.xxx.xxx TeleStaxAS
   127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
   ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
   ```

   Note: You will have to be a root to have privileges to do modify the *hosts* file.

   Note: The hostname should match exactly what is returned by executing "hostname" from the command prompt.

   Note: This must be the first line in the */etc/hosts* file. If not, you might encounter a "503 Service Unavailable" error.

6. To activate the changes made to the *hosts* file, run the following command at the prompt:

   ```
   service network restart
   ```

Firewall Configuration

Several ports must be allowed to go through the firewall. Ports that are going to be in use are 8080, 9990, 5080, and the optional remote debugging port 8787. To do this, edit the */etc/sysconfig/iptables* file:

```
vi /etc/sysconfig/iptables
```

Then, add the ports as illustrated below:

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 9990 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5080 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 5080 -j ACCEPT
```

```
#optional port needs to be opened if platform remote debugging will be used.
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8787 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Save the file and restart the firewall for the changes to take effect by executing the following command:

```
service iptables restart
```

TeleStax Installation

There are many ways to install the platform, but for this example, the user "jboss" is created and used. The example "jboss" user account will be used throughout the installation process as a home directory for the platform.

After creating the "jboss" user account in the system, log in to that account. Then, copy the desired platform distribution to the user home directory. In this example, the user home directory is */home/jboss*.

Copy the following to the system under the */home/jboss* directory and unzip them.

```
(TelScale AS) - TelScale-SIP-Servlets-7.0.3.329-jboss-as-7.2.0.Final.zip
(Mobicents AS) - mss-3.1.633-jboss-as-7.2.0.Final.zip
```

TeleStax Configuration

To properly configure the newly installed platform in a system, specific modifications of the system configuration need to take place.
Edit the *standalone-sip.xml* file:

```
${JBOSS_HOME}/standalone/configuration/standalone-sip.xml
```

Locate the lines that need to be modified. They are identified below in **RED** in the "Before" section. Replace the lines in **RED** in the "Before" section with the lines in **RED** from the "After" section.

Before

```
        <subsystem xmlns="urn:jboss:domain:webservices:1.2">
            <modify-wsdl-address>true</modify-wsdl-address>
            <wsdl-host>${jboss.bind.address:127.0.0.1}</wsdl-host>
            <endpoint-config name="Standard-Endpoint-Config"/>
            <endpoint-config name="Recording-Endpoint-Config">
….
    <interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:127.0.0.1}"/>
        </interface>
        <!-- TODO - only show this if the jacorb subsystem is added  -->
        <interface name="unsecure">
            <!--
```

```
            ~  Used for IIOP sockets in the standard configuration.
            ~                  To secure JacORB you need to setup SSL
            -->
            <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
        </interface>
    </interfaces>
```

After

```
        <subsystem xmlns="urn:jboss:domain:webservices:1.2">
            <modify-wsdl-address>true</modify-wsdl-address>
            <wsdl-host>${jboss.bind.address: TeleStaxAS}</wsdl-host>
            <endpoint-config name="Standard-Endpoint-Config"/>
            <endpoint-config name="Recording-Endpoint-Config">
….
    <interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management: TeleStaxAS}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address: TeleStaxAS}"/>
        </interface>
        <!-- TODO - only show this if the jacorb subsystem is added  -->
        <interface name="unsecure">
            <!--
              ~  Used for IIOP sockets in the standard configuration.
              ~                  To secure JacORB you need to setup SSL
              -->
            <inet-address value="${jboss.bind.address.unsecure: TeleStaxAS}"/>
        </interface>
```

TeleStax Startup

To run the Application Server, go to the following directory:

```
${JBOSS_HOME}/bin
```

Then, execute the following command:

```
./standalone.sh -c standalone-sip.xml
```

```
================================================================================
==                                                                            ==
==              Thank you for running Mobicents Community code                ==
==   For Commercial Grade Support, please request a TelScale Subscription     ==
==                         http://www.telestax.com/                           ==
==                                                                            ==
================================================================================

2014-04-22 11:59:44,936 WARN  [SipStackImpl] (main) Could not register the stack as a Notification L
istener of jboss.system:service=Logging,type=Log4jService runtime changes to log4j.xml won't affect
SIP Stack Logging
Apr 22, 2014 11:59:44 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 23281 ms
```

To stop the service, press **Ctrl**+**C**.

TeleStax Verification

Once the Application Server service is started, access to the TeleStax JBoss Web Administration Console can be done from any browser by going to the following URL:

http://<as_ip_address>:8080

By default, you will not be able to access the Administration Console. If you try to access it you will see the following screen:

By default the realm name used by AS 7 is "ManagementRealm" this is already selected by default.

After you have added the user follow this link to Try Again.

Because no user access has been set up yet, follow these steps to enable user access:

Go to the */opt/mss-3.1.620-jboss-as-7.2.0.Final/bin/* directory, and execute the following command:

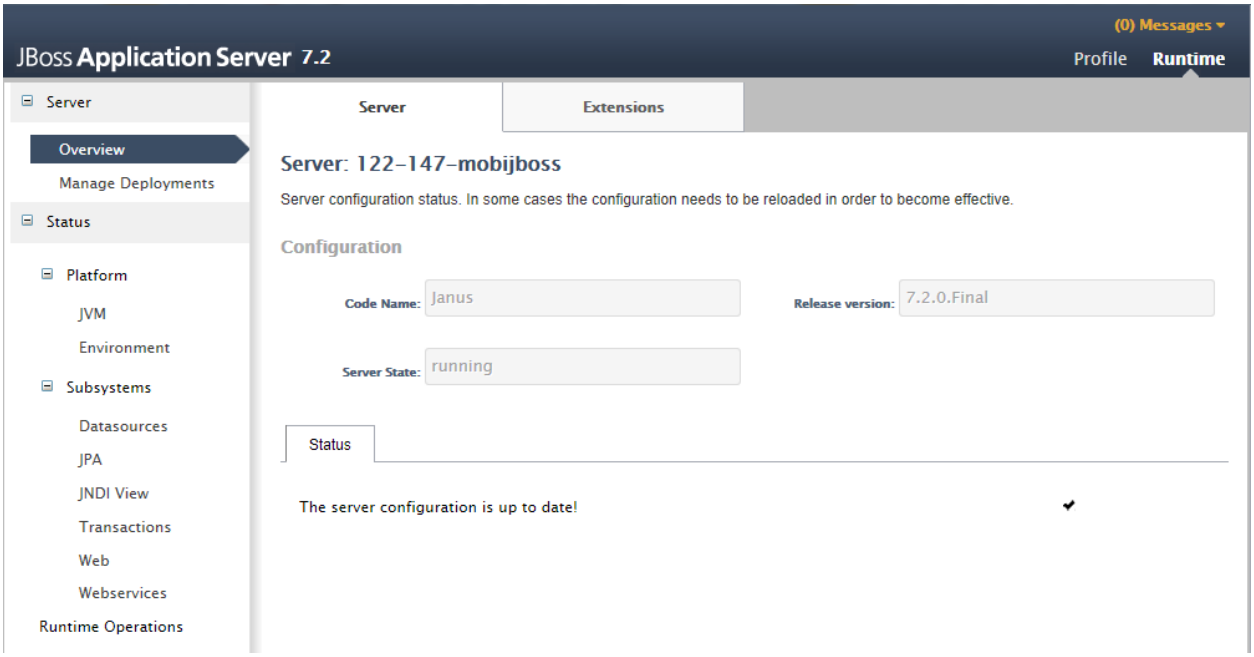
```
./add-user.sh
```

Follow the prompts:

1. Select **Management User** by pressing **Enter**.
2. Press **Enter** to accept **ManagementRealm**.
3. Enter a username of your choice (for example, admin).

   Note: You will need these credentials to log in to the Web Console later on.
4. Enter a password and confirm it (for example, testing1!).
5. Respond "yes" to the next three questions and user access setup is complete.
6. Navigate to http://<as_ip_address>:8080 and click **Administration Console**.
7. Enter the log in credentials you created and click **Log In**.

Note: You can always re-run *add-user.sh* script to change previously defined settings.



8. Appendix B: Updating the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is a set of the three files as described by the following section: Dialogic JSR 309 Connector Requirements

In the TeleStax Application Server JBoss version, the required application files are part of the application WAR file structure.

In the application WAR file the previous versions of connector JAR files need to be removed and replaced with new versions:

```
<application_WAR_File>/WEB-INF/lib/Dialogic309-5.#.####-jboss.jar
<application_WAR_File>/WEB-INF/lib/Dialogic309msmltypes-5.#.####-jboss.jar
<application_WAR_File>/WEB-INF/lib/Dialogic309smiltypes-5.#.####-jboss.jar
```