



Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 5.2

**Installation and Configuration Guide
with Oracle Communications Converged Application Server 7**

Copyright and Legal Notice

Copyright © 2016 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

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
For Developers	7
3. Installation and Configuration	8
Preparing the J2EE Converged Application Server	8
Installing the Dialogic JSR 309 Connector	8
Configure Application Server Platform	8
Install the Dialogic JSR 309 Connector	10
Installing the Dialogic JSR 309 Verification Application	19
Preparing for the Dialogic JSR 309 Verification Application	19
Deploying the Dialogic JSR 309 Verification Application	21
Configuring the Platform's SIP Router	27
Configuring the PowerMedia XMS Media File.....	29
Running the Dialogic JSR 309 Verification Application.....	31
4. Dialogic JSR 309 Verification Application	32
About.....	32
The Details	32
Application WAR File Content	32
Application Initialization	33
Mapping SIP Traffic to the Application	35
Dialogic JSR 309 Connector Initialization	35
5. Troubleshooting	37
Logging.....	37
Dialogic JSR 309 Connector and Verification Application Troubleshooting	38
SIP Errors.....	39
6. Building and Debugging Verification Application in Eclipse IDE	40
Prerequisites.....	40
Creating the Build Environment.....	41
Importing the Project from Distribution	41
Configuring the Project.....	46
Building the Project.....	61
Configuring Eclipse Project and OCCAS Deployed Application for Remote Debugging	62
Eclipse Project Remote Debugging Configuration	63
7. Appendix A: Dialogic JSR 309 Connector Environment Setup	65
About.....	65
Installing and Configuring OCCAS	65
Preinstallation Setup	65
OCCAS Installation	66
OCCAS Configuration	78
OCCAS Startup.....	86
Firewall Configuration.....	86
OCCAS Verification	87
8. Appendix B: Updating the Dialogic JSR 309 Connector	88

Revision History

Revision	Release Date	Notes
2.0	July 2016	Updates for JSR 309 5.2 Service Update 1. Contents of the Distribution : Added the For Developers section. Installation and Configuration : Updated the Configuring the Platform's SIP Router script. Dialogic JSR 309 Verification Application : Updated the Application Initialization section. Logging : Updated the script.
1.0	March 2016	Initial release of this document.
Last modified: July 2016		

1. Dialogic JSR 309 Connector Requirements

The following requirements are needed before the J2EE application can use the Dialogic JSR 309 Connector:

- A functional Oracle Communications Converged Application Server 7.0 (OCCAS 7.0) platform for development and testing.
 - The Dialogic JSR 309 Connector has been tested with the OCCAS version 7.0 Application Server.
- A functional PowerMedia XMS Release 3.1 system.
- SIP phones or soft clients, or WebRTC client (Chrome or Firefox web browser).

Note: WebRTC is not supported as JBoss platform does not support websocket Contents of the Distribution.

OCCAS7.0 Application Server Platform	Dialogic JSR 309 Connector
occas_generic.jar	dialogic309-5.x.xxxx-occas7.0.tar

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-occas7.0.tar

Where:

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

Note: OCCAS7.0 connector is built with Java version 1.8.

This package contains the following structure.

Dialogic JSR 309 Connector Files	Description
<u>DIR:</u> <i>/DlgcJSR309/application/</i> <u>CONTENTS:</u> <i>dlgc_sample_demo.war</i> <i>Dialogic.mp4</i> <i>Project/</i>	Directory that contains Dialogic JSR 309 Verification Application <i>dlgc_sample_demo.war</i> ready to be deployed and the <i>Dialogic.mp4</i> 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 <i>dlgc_sample_demo.war</i> . Refer to Dialogic JSR 309 Verification Application for details.
<u>DIR:</u> <i>/DlgcJSR309/dialogic309Connector/</i> <i>/DlgcJSR309/3rdPartyLibs/</i>	Directory that contains the <i>Dlgc309Connector</i> , which has all of the Dialogic connector files, and the <i>3rdPartyLibs</i> directory, which has all necessary third-party JAR files.
<u>DIR:</u> <i>/DlgcJSR309/properties/</i> <u>CONTENTS:</u> <i>dlgc_sample_demo.properties</i> <i>log4j2.xml</i>	Directory that contains Verification Application properties files used to set up its configuration and the configuration parameters for Dialogic JSR 309 Connector. Directory also contains the <i>Log4j2.xml</i> log configuration file used for Dialogic JSR 309 Connector and Verification Application logging.

For Developers

To make it easier for developers, Dialogic provides connector files for direct download from an HTTPS URL.

- <https://www.dialogic.com/files/jsr-309/5.2SU1/dialogic309-5.2-SU1-1102-occas7.0.jar>
- <https://www.dialogic.com/files/jsr-309/5.2SU1/dialogic309-5.2-SU1-1102-occas7.0.war>
- <https://www.dialogic.com/files/jsr-309/5.2SU1/dialogicmsmltypes-5.2-SU1-1102.jar>
- <https://www.dialogic.com/files/jsr-309/5.2SU1/dialogicsmiltypes-5.2-SU1-1102.jar>

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. This section contains the following procedures:

1. [Preparing the J2EE Converged Application Server](#)
2. [Installing the Dialogic JSR 309 Connector](#)
3. [Installing the Dialogic JSR 309 Verification Application](#)
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 OCCAS 7 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 a library where, once properly installed and configured, it can be used by an application.

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

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

1. [Configure the Application Server Platform](#)
2. [Install the Dialogic JSR 309 Connector](#)

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 contents in detail.

Configure Application Server Platform

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

```
tar -xvf dialogic309-x.x.xxxx-occas7.0.tar
```

This will create the *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:

- [Set Up the Platform's Environment](#)
- [Set Up and Configure the Logging Facility](#)

Set Up the Platform's Environment

Edit the *startWeblogic.sh* file located here:

```
${DOMAIN_HOME}/bin/startWeblogic.sh
```

Add the lines in **RED** at the appropriate locations as illustrated below, and then save the changes.

```
....
SAVE_JAVA_OPTIONS="${JAVA_OPTIONS}"

# Dialogic additions
LOG4J_OPTIONS="-Dlog4j.configurationFile=${DOMAIN_HOME}/config/Dialogic/log4j2.xml"
CLASSPATH="${DOMAIN_HOME}/lib/log4j-api-2.2.jar:${DOMAIN_HOME}/lib/log4j-slf4j-impl-2.2.jar:${DOMAIN_HOME}/lib/log4j-core-2.2.jar:${CLASSPATH}"
SERIALIZATION="-Dwlss.local.serialization=false"
DLGC_OPTS="${SERIALIZATION} ${LOG4J_OPTIONS}"
# END Dialogic additions

SAVE_CLASSPATH="${CLASSPATH}"
....

if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
    echo "Starting WLS with line:"
    echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DLGC_OPTS} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} ${PROXY_SETTINGS} ${SERVER_CLASS}"
    ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DLGC_OPTS} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} ${PROXY_SETTINGS} ${SERVER_CLASS}
else
    echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
    ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DLGC_OPTS} -Djava.security.policy=${WLS_POLICY_FILE} ${JAVA_OPTIONS} ${PROXY_SETTINGS} ${SERVER_CLASS}
    >"${WLS_REDIRECT_LOG}" 2>&1
fi
....
```

Set Up and Configure the Logging Facility

The log4j2 logging facility is implemented through five third-party log4j2 library files (JAR files), which need to be placed in the appropriate platform *lib* directory. From 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:

```
${DOMAIN_HOME}/lib
```

From the Dialogic JSR 309 Connector distribution, copy *log4j2.xml* into the following platform directory:

```
"${DOMAIN_HOME}/config/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:

```
"${DOMAIN_HOME}/logs/Dialogic.log"
```

- Default logging configuration is set to ERROR. Configuration file *Log4j2.xml* can be edited 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.

Install the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is distributed as set of files. The set includes the following files:

- A connector WAR file:
 - *dialogic309-5.x.xxxx-occas7.0.war*
- Set of connector library (JAR) files:
 - *dialogic309-5.x.xxxx-occas7.0.jar*
 - *dialogicsmiltpes-5.x.xxxx.jar*
- Set of required third-party library (JAR) files:
 - *geronimo-commonj_1.1_spec-1.0.jar*
 - *jain-sip-ri-1.2.256.jar*
 - *xbean.jar*

Follow these steps to properly configure Dialogic JSR 309 Connector:

1. [Configuring the Dialogic JSR 309 Connector Required Library \(JAR\) Files](#)
2. [Deploying the Dialogic 309 Connector WAR File](#)
3. [Configuring the Media Server Control](#)

Configuring the Dialogic JSR 309 Connector Required Library (JAR) Files

From distribution, copy the following library (JAR) files:

- *dialogic309-5.x.xxxx-occas7.0.jar*
- *dialogic309smiltpes-5.x.xxxx-occas7.0.jar*
- *geronimo-commonj_1.1_spec-1.0.jar*
- *jain-sip-ri-1.2.256.jar*
- *xbean.jar*

Place the files in the following folder:

`${DOMAIN_HOME}/lib`

Note: *DOMAIN_HOME* refers to the domain path as specified during OCCAS installation.

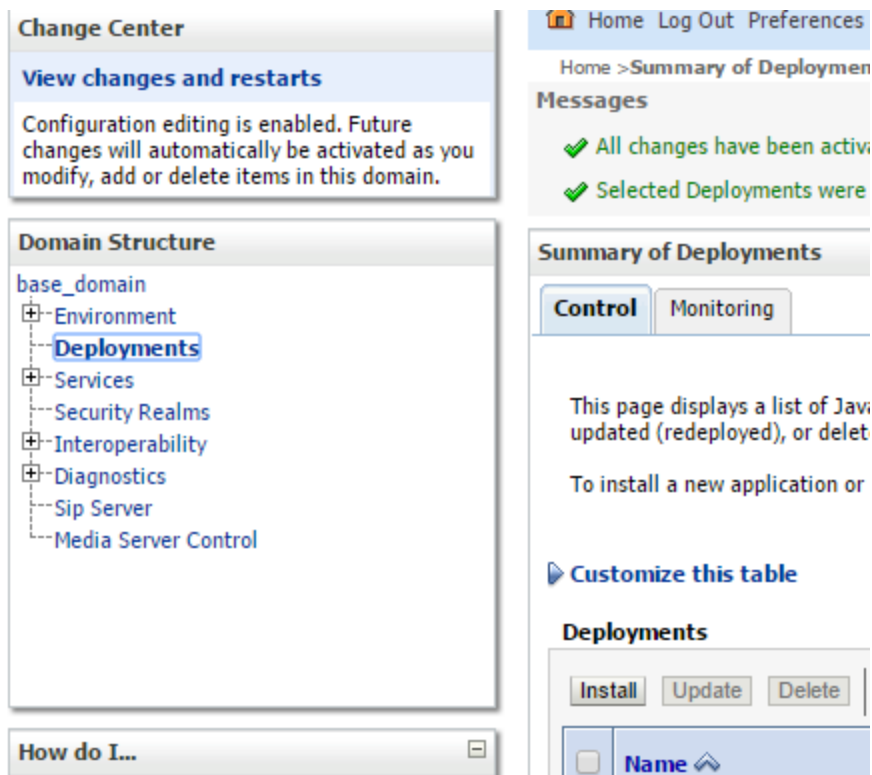
Deploying the Dialogic 309 Connector WAR File

From distribution, copy the *dialogic309-5.x.xxxx-occas7.0.war* file to the OCCAS `<DOMAIN_HOME>/applications` directory.

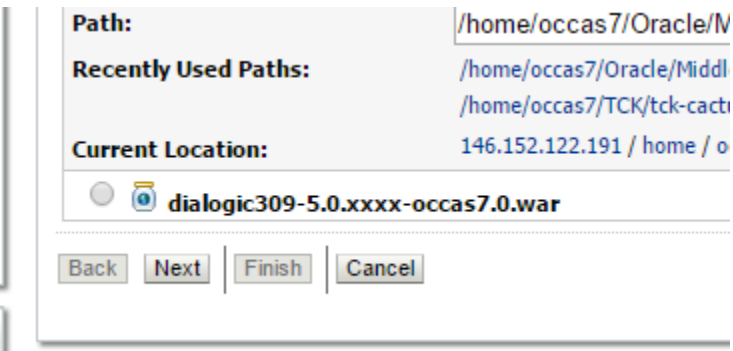
Start the OCCAS Application Server.

Access the Administration Console and log in with the appropriate credentials.

1. In the Administration Console, click **Deployments** in the **Domain Structure** section.



2. In the **Summary of Deployments** section, click **Install**, select *dialogic309-5.x.xxxx-occas7.0.war*, and then click **Next**.



3. Select **Install this deployment as an application**, and then click **Next**.

5. Click **Finish**.

Install Application Assistant

Back

Next

Finish

Cancel

Review your choices and click Finish

Click Finish to complete the deployment. This may take a few moments to complete.

Additional configuration

In order to work successfully, this application may require additional configuration. Do you want to

☒ **Yes, take me to the deployment's configuration screen.**

☐ **No, I will review the configuration later.**

Summary

Deployment: /home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base

Name: dialogic309-5.0.xxxx-occas7.0

Staging Mode: I will make the deployment accessible at /home/occas7/Oracle/Middleware/O
5.0.xxxx-occas7.0.war

Plan Staging Mode: Use the same accessibility as the application

Security Model: DDOnly: Use only roles and policies that are defined in the deployment descri

Target Summary

Components ^

dialogic309-5.0.xxxx-occas7.0

Back

Next

Finish

Cancel

After some time you will be directed to the following summary screen. It is important that **Deployment Order** is set to a lower order number than any application using the Dialogic JSR 309 Connector. The **Deployment Order** is set to 100 by default.

Settings for dialogic309-5.0.xxxx-occas7.0

Overview Deployment Plan Configuration Security Targets Control Testing Monitoring Notes

Save

Use this page to view the installed configuration of a Web application.

Name:

dialogic309-5.0.xxxx-occas7.0

Context Root:

/dialogic309-5.0.xxxx-occas7.0

Path:

/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/applications/dialogic309-5.0.xxxx-occas7.0.war

Deployment Plan:

(no plan specified)

Staging Mode:


nostage


Plan Staging Mode:

(not specified)

Security Model:

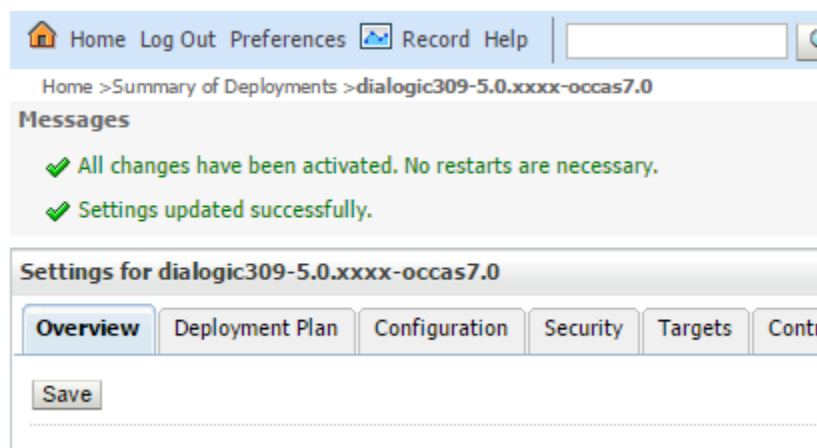
DDOnly

 **Deployment Order:**

 **Deployment Principal Name:**

Save

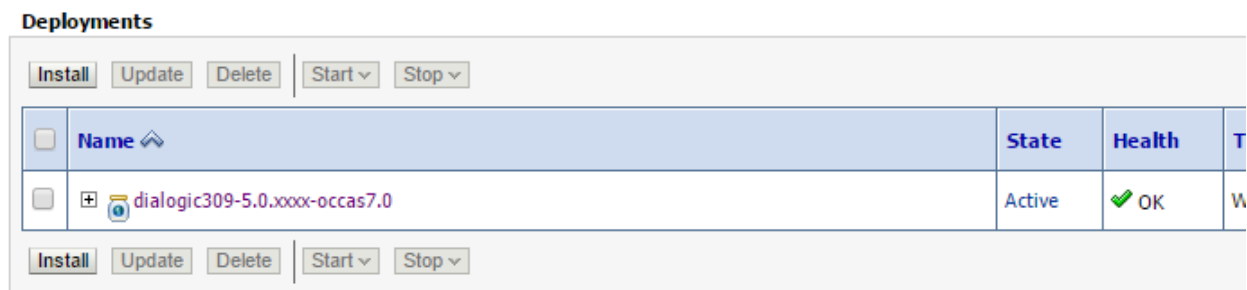
When finished, click **Save**. If configuration was successful, two messages will appear as shown in the image:



To make sure that Dialogic connector started properly click **Deployments** in the **Domain Structure** section.



If configured correctly, the Dialogic JSR 309 Connector should show **State** as **Active** and **Health** as **OK**.



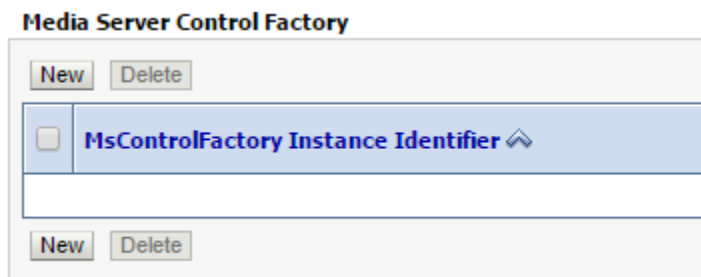
Configuring the Media Server Control

The Dialogic JSR 309 Connector needs to be configured via **Media Server Control** in order for the application to be able to access it.

1. Click on **Media Server Control** in the **Domain Structure** section.



2. Click **New**.



3. Specify the following parameters, and then click **Next**:

- **MsControlFactory Instance Identifier** - This can be any string (for example, DlgcJsr309Factory).
- **JNDI Name** - This can be any string (for example, DlgcJsr309Factory).
- **Driver Name** - With Dialogic JSR 309 Connector started correctly, select **com.dialogic.dlg309**.

Create the Media Server Control Factory

Create Media Server Control MsControlFactory Instance Identifier, JNDI Name and Driver Name.

In this step, please input the MsControlFactory Instance Identifier, JNDI name and driver name.

* Indicates required fields

What would you like to name your new MsControlFactory Instance Identifier?

* **MsControlFactory Instance Identifier:**

What JNDI name would you like to assign to your new MsControlFactory Instance Identifier?

* **JNDI Name:**

What driver name would you like to select?

* **Driver Name:**

4. In the **Configure Properties** section, observe the various parameters that are used to configure Dialogic JSR 309 Connector interface.

Create the Media Server Control Factory

Back | Next | Finish | Cancel

Modify the properties.

The following properties will be used to create your new factory.

* Indicates required fields

What property would you like modify, delete or add?

*** Configure Properties:**

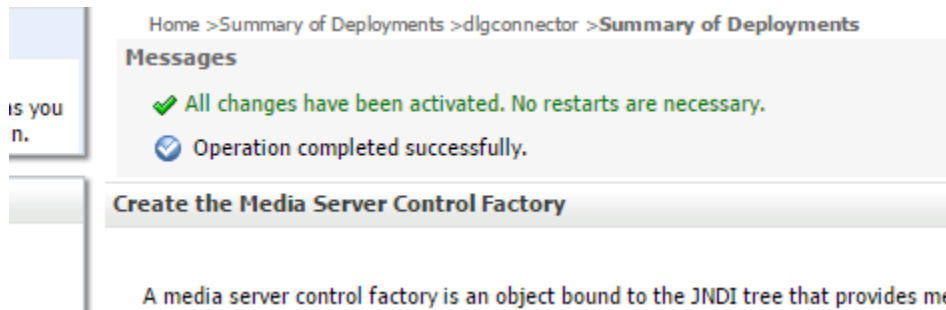
```
mediaserver.msType = XMS
connector.sip.address = 192.168.1.100
mediaserver.sip.ipaddress = xxx.xxx.xxx.xxx
mediaserver.sessionTimer.maxTimeout = 1800
connector.sip.transport = udp
mediaserver.sip.port = 5060
version = 5.0
connector.sip.port = 5060
mediaserver.sessionTimer.switch = on
APPSERVER_PLATFORM = OCCAS7
mediaserver.sessionTimer.minTimeout = 100
connector.conferenceControlLeg = yes
```

Back | Next | Finish | Cancel

Note the following parameters:

- *connector.sip.address* – This field is automatically discovered when Dialogic Connector starts up. Connector initialization logic picks up the IP address associated with the first network interface available in the platform. If the platform has multiple network interfaces configured, it might be necessary to adjust this IP address to an appropriate one.
- *connector.sip.port* – This field is automatically discovered when Dialogic Connector starts up. Connector initialization logic picks up the port associated to the first network interface available in the platform. If the platform has multiple network interfaces configured, it might be necessary to adjust this port address to an appropriate one.
- *mediaserver.sip.ipaddress* and *mediaserver.sip.port* – Specify the Dialogic PowerMedia XMS Media Server appropriate IP address and IP Port. Replace "xxx.xxx.xxx.xxx" with the appropriate IP address. Note that the port is automatically specified as 5060 because it is the default port on XMS.

- Click **Finish**. If the configuration was successful, a message will appear as shown in the image.



It is important to note that the **JNDI Name** content is what the application will need to use when configuring itself to use the Dialogic JSR 309 Connector factory.

JNDI Name	
mscontrol/DlgcJsr309Factory	

The Dialogic JSR309 Connector is now configured and ready to be used by the application.

Installing the Dialogic JSR 309 Verification Application

Distribution package comes with a sample Verification Application which uses Dialogic JSR 309 connector. This section describes how to install, configure, and run sample application using Dialogic XMS Media Server via JSR 309 API.

There are several components that make up the Dialogic JSR 309 Verification Application:

- Required third-party libraries (JAR files)
- Verification Application (*dlg_sample_demo.war*)
- Verification Application configuration file (*dlg_sample_demo.properties*)

Installing the Dialogic JSR 309 Verification Application requires three steps:

1. [Preparing for the Dialogic JSR 309 Verification Application](#)
2. [Deploying the Dialogic JSR 309 Verification Application](#)
3. [Configuring the Platform's SIP Router](#)

Preparing for the Dialogic JSR 309 Verification Application

Prepare for installing the Dialogic JSR 309 Verification Application:

1. Copy the distribution package *DlgcJSR309/3rdPartyLibs/json_simple-1.1.jar* library file the platform's library directory:

```
${DOMAIN_HOME}/lib
```

2. Copy the distribution package *DlgcJSR309/application/dlgc_sample_demo.war* verification demo WAR file to the platform's designated application directory:

```
${DOMAIN_HOME}/applications
```

3. Copy the distribution package *DlgcJSR309/properties/dlgc_sample_demo.properties* properties file to the platform's designated configuration directory:

```
${DOMAIN_HOME}/config/Dialogic" directory
```

Note: The properties file contains various settings used by the application. For detailed explanation of various parameters refer to the description in section of this documentation.

Before the application can be deployed, the path to application's configuration file (*dlgc_sample_demo.properties*) must be made accessible to the application. One way to accomplish this is to export its path in its environment. To do so, edit user's .bashrc file and add the following lines in **RED**:

```
# .bashrc
export JAVA_HOME=/usr/java/jdk1.8.0_65

# Dialogic Additions:
export DOMAIN_HOME=/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain
export SAMPLE_PROPERTY_FILE=${DOMAIN_HOME}/config/Dialogic/dlgc_sample_demo.properties
# 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:

- DOMAIN_HOME points to a root directory of the installed platform's domain
- SAMPLE_PROPERTY_FILE points to the configuration properties file used by Dialogic JSR 309 verification demo.

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

In order for any changes in .bashrc file to take an effect, the user either needs to log out and log back in or execute the "source" command from user's home directory:

```
source /home/occas7/.bashrc
```

Deploying the Dialogic JSR 309 Verification Application

To deploy the Verification Application (WAR file) in a platform, start the platform by executing following command:

```
${DOMAIN_HOME}/bin/startWebLogic.sh
```

Access the Administration Console through the web browser at:

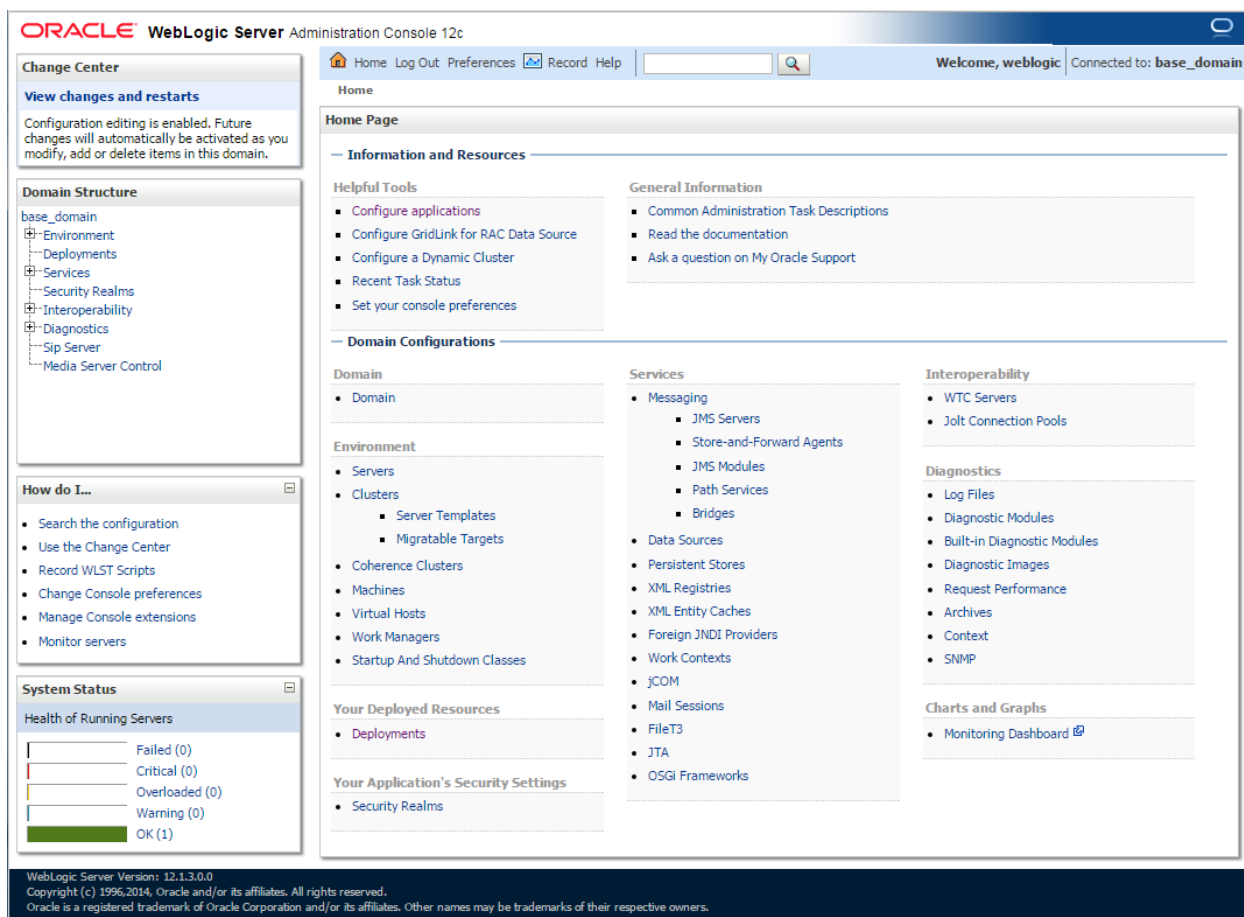
http://<AS_ip_address>:7001/console

Use the username/password as set during configuration. The following credentials were set during installation of the platform in Appendix A:

Username: weblogic

Password: WebLogic!! ("0" is a zero)

Note: The password is as defined during OCCAS 7 installation.

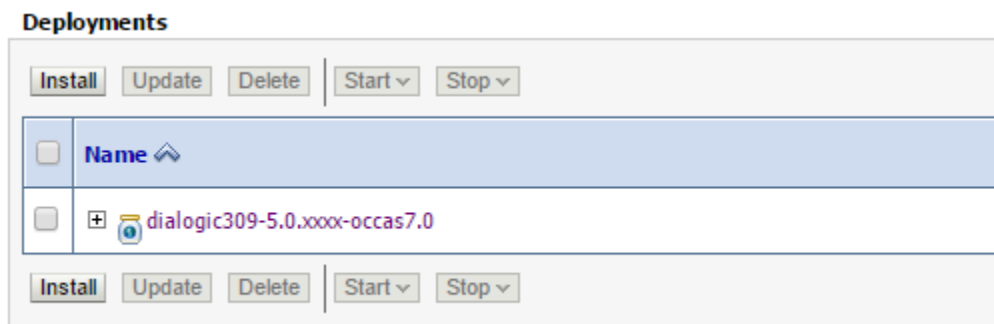


1. In the Administration Console, click **Deployments** in the **Domain Structure** section.

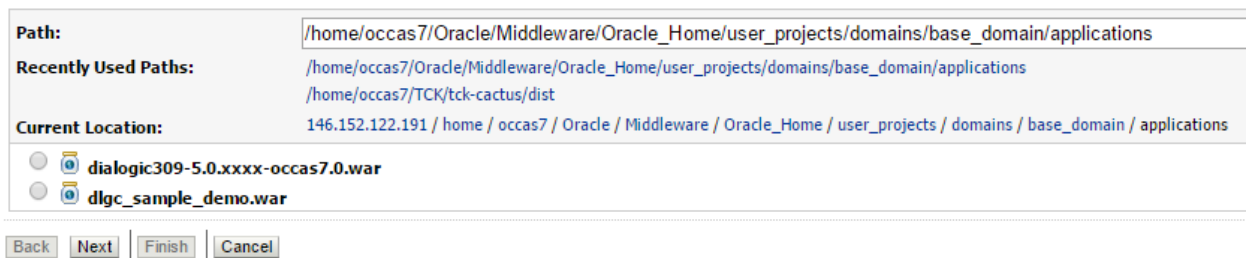


Note: If OCCAS was installed in **Production Mode**, click **Lock & Edit** and make the necessary changes. For the changes to take effect, click **Release Configuration**. When installed in **Development Mode**, this process is suppressed. The examples in this section are based on OCCAS 7 being installed in **Development Mode**.

2. Click **Install**.



3. In the **Current Location** field, navigate to *<Domain Location>/applications*, select **dlgc_sample_demo.war**, and then click **Next**.



4. Select **Install this deployment as an application**, and then click **Next**.

Install Application Assistant

Back Next Finish Cancel

Choose targeting style

Targets are the servers, clusters, and virtual hosts on which this deployment is targeted. The application and its components will be targeted to the same locations.

☒ **Install this deployment as an application**

The application and its components will be targeted to the same locations.

☐ **Install this deployment as a library**

Application libraries are deployments that are available for other deployments.

Back Next Finish Cancel

5. Select **I will make the deployment accessible from the following location**, and then click **Next**.

Source Accessibility

How should the source files be made accessible?

☐ **Use the defaults defined by the deployment's targets**

Recommended selection.

☒ **I will make the deployment accessible from the following location**

During deployment, the files will be copied automatically to the Managed Servers to which the application is targeted.

Location:

6. Click **Finish**.

Install Application Assistant

Back

Next

Finish

Cancel

Review your choices and click Finish

Click Finish to complete the deployment. This may take a few moments to complete.

Additional configuration

In order to work successfully, this application may require additional configuration. Do you want to review this application's configuration?

☒ **Yes, take me to the deployment's configuration screen.**

☐ **No, I will review the configuration later.**

Summary

Deployment: /home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/application

Name: dlgc_sample_demo

Staging Mode: I will make the deployment accessible at
/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/application

Plan Staging Mode: Use the same accessibility as the application

Security Model: DDOnly: Use only roles and policies that are defined in the deployment descriptors.

Target Summary

Components ⌵

dlgc_sample_demo

Back

Next

Finish

Cancel

6. Click **Save**.

Settings for dlgc_sample_demo

Overview Deployment Plan Configuration Security Targets Control Testing Monitoring Notes

Save

Use this page to view the installed configuration of a Web application.

Name:

dlgc_sample_demo

Context Root:

/dlgc_sample_demo

Path:

/ home/ occas7/ Oracle/ Middleware/ Oracle_Home/ user_projects/ domains/ base_domain/ applications/ dlgc_sample_demo. war

Deployment Plan:

(no plan specified)

Staging Mode:


nostage


Plan Staging Mode:

(not specified)

Security Model:

DDOnly

 **Deployment Order:**

 **Deployment Principal Name:**

Save

If deployment was successful, two messages will appear as shown in the image.

Messages

- ✓ All changes have been activated. No restarts are necessary.
- ✓ Settings updated successfully.

Settings for dlgc_sample_demo

Overview | Deployment Plan | Configuration | Security | Targets | Control | Testing | Monitoring | Notes

Use this page to view the installed configuration of a Web application.

Under **Deployments**, the newly deployed *dlgc_sample_demo* application state and health are displayed. The application should be in an active state.

Deployments

|

<input type="checkbox"/>	Name ↕	State	Health	Type
<input type="checkbox"/>	 dialogic309-5.0.xxxx-occas7.0	Active	✓ OK	Web App
<input type="checkbox"/>	 dlgc_sample_demo	Active	✓ OK	Web App

|

Note: Selecting a deployment provides options to *Stop*, *Start*, or *Delete* it.

Configuring the Platform's SIP Router

In order for the application server platform to correctly route incoming SIP traffic to the Verification Application, the following SIP router configuration settings are required.

In the following platform directory, create the following file:

```
${DOMAIN_HOME}/approuter/approuter.json
```

Edit the file and paste the following content in it exactly as shown below:

```
{
  "chains" :
  [
    {
      "description" : "Dialogic Demo Application Chain",
      "criteria" :
      {
        "or" :
        [
          {
            "and" :
            {
              "equal" :
              {
                "request.method" : "INVITE"
              },
              "contains":
              {
                "ignore-case" : "true",
                "request.uri.user" : "player"
              }
            }
          },
          {
            "and" :
            {
              "equal" :
              {
                "request.method" : "INVITE"
              },
              "contains" :
              {
                "ignore-case" : "true",
                "request.uri.user" : "pattern"
              }
            }
          }
        ]
      }
    }
  ],
}
```

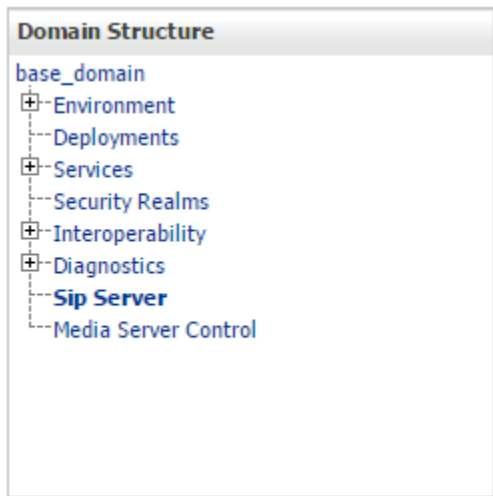
```

    "applications" :
    [
        {
            "name" : "DialogicSampleDemo",
            "subscriber": "request.from",
            "region":"TERMINATING",
            "route-modifier":"NO_ROUTE"
        }
    ]
}

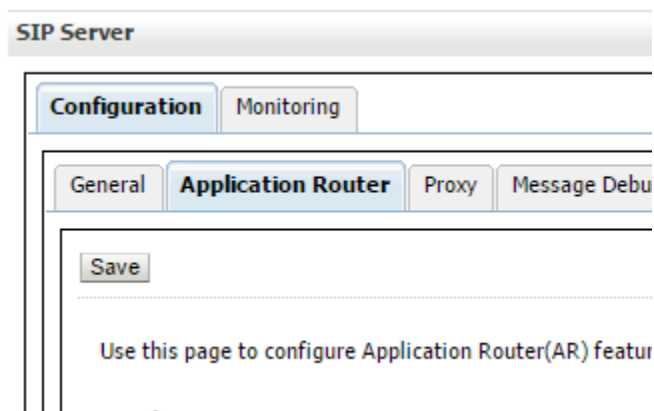
```

Note: The above approuter.json content assumes that no previous SIP routing rules have been set. For more details on creating json SIP routing rules, refer to OCCAS 7.0 documentation.

1. In the Administration Console, click **Sip Server** in the **Domain Structure** section.





2. Click the **Application Router** tab.



3. In the **Default Application Router** section, select **Use Json form configuration file** and enter the filename **approuter.json** (as created earlier).

Default Application Router

☒  Use Json form configuration file

 Json filename:

Custom Application Router

Now, the above defined .json filename needs to be created or edited if already exists. This file should be found in the `${DOMAIN_HOME}/approuter` folder. Edit `approuter.json` and put the appropriate content inside of it.

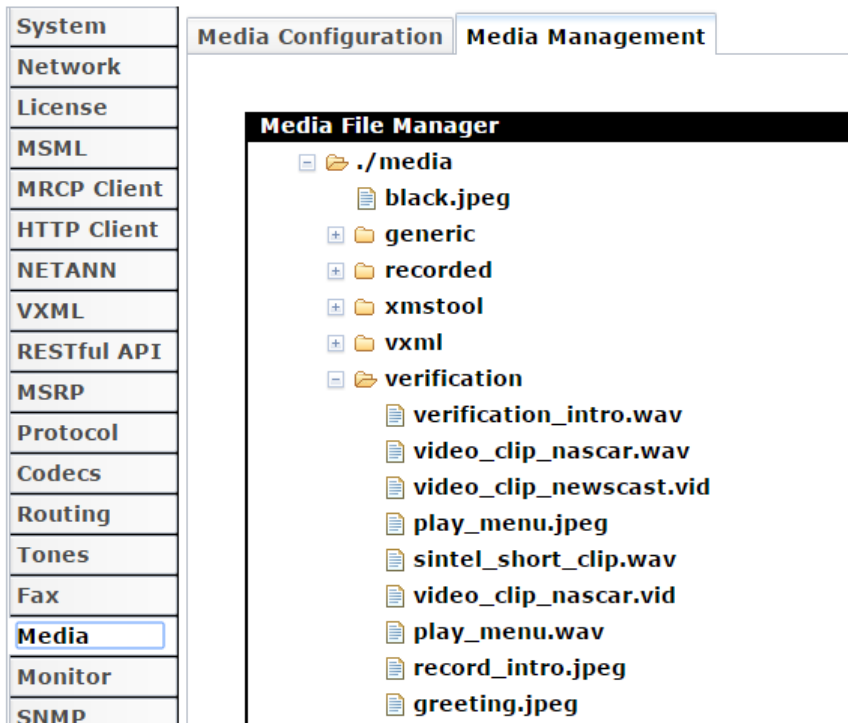
Configuring the PowerMedia XMS Media File

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.

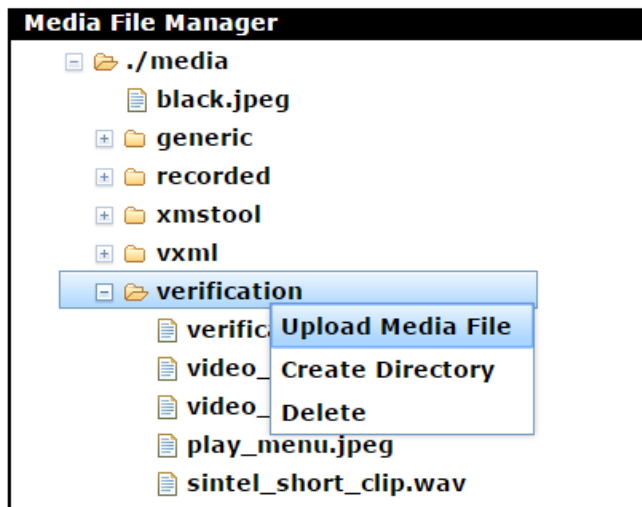
1. To install *Dialogic.mp4* manually, log in to PowerMedia XMS WebGUI, and then click **Media**.

System	General	Services	Time	Backup/Restore	Upgrade	NFS Mount Points	Maintenance	Account Manager
Network	XMS							
License	release		3.0.11915					
MSML	state		RUNNING					
MRCP Client	System							
HTTP Client	os release		CentOS release 6.7 (Final)					
NETANN	os version		Linux 2.6.32-573.7.1.el6.x86_64					
VXML	uptime		16 days 22 hours 27 minutes 4 seconds					
RESTful API	cpu load		T1=0.01 , T5=0.04 , T15=0					
MSRP	memory		total:3913424 KB used:2091660 KB					
Protocol	System Storage							
Codecs	/dev/mapper/vg_12 lv_root (/)		total: 51475068 KB, used: 9733092 KB					
Routing	/dev/sda1 (/boot)		total: 487652 KB, used: 130413 KB					
Tones	/dev/mapper/vg_12 lv_home (/home)		total: 182089932 KB, used: 60744 KB					
Fax	System Time							
Media	time		Fri Jan 22 15:14:52 2016					
Monitor	zone							
SNMP								

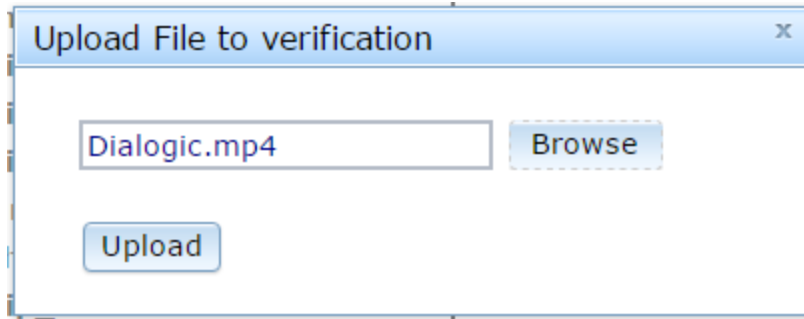
2. Click the **Media Management** tab.



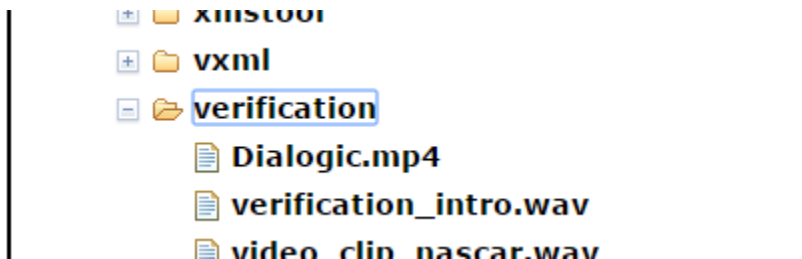
3. In the **Media File Manager** section, Right-click the **verification** folder and click **Upload Media File**.



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 309 Verification Application is a simple application that can be used for proper operation of the platform and the Dialogic 309 Connector. On successful connection, the Verification Application will play a sample *Dialogic.mp4* audio/video file. The application adjusts the play based on client capabilities. Therefore, if the client supports audio only connection, then only the audio portion of the sample .mp4 file will be played.

The Verification Application is written to accept either a SIP or web browser endpoints.

Perform the following procedure for SIP client verification:

1. Configure a SIP client for the supported audio/video codec.
2. Call into the OCCAS 7 Application Server with following URI:

```
player@<AS_ip_address>
```

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

To perform web browser client verification (Chrome and Firefox only), open a Chrome or Firefox web browser and navigate to the following URL:

```
<AS_IP_ADDRESS>:7001/dlgc_sample_demo/DemoGUI/index.html
```

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 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.

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](#)
- [Mapping SIP Traffic to the Application](#)
- [Dialogic JSR 309 Connector Initialization](#)

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 */DemoGUI* directory contains content for the application and supports WebRTC. It includes necessary .html and .js files for the verification demo.
- 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.
 - Contains the *web.xml* file used for the web side of the application WAR.

Application Initialization

In Oracle OCCAS7 platform, we make a full use of SipApplication annotations. However, with such an approach, a verification application containing more than one SipServlet needs to take advantage of servlet dispatcher logic. Therefore, in the application there is a MainServlet.java that frontends all SIP Invites and dispatches them to the appropriate SipServlet:

```
@javax.servlet.sip.annotation.SipApplication(name = "DialogicSampleDemo", sessionTimeout = 30,
distributable = true, mainServlet = "MainServlet")

@javax.servlet.sip.annotation.SipServlet(name = "MainServlet", applicationName =
"DialogicSampleDemo", loadOnStartup = 0, description = "MainServlet description...")

public class MainServlet extends SipServlet {

    private static final long serialVersionUID = 1L;

    public static Logger          log          =
    LoggerFactory.getLogger(PatternDetection.class);

    @Override
    public void doInvite(javax.servlet.sip.SipServletRequest req) throws
    javax.servlet.ServletException, IOException {
        SipURI reqUri = (SipURI) req.getRequestURI();
        String user = reqUri.getUser();
        log.debug("Entering");
        log.info("From: " + req.getFrom().toString());
        if (user != null) {
            RequestDispatcher dispatcher = null;
            if (user.equals("player")) {
                log.debug("Invite for: " + user);
                dispatcher = getServletContext().getNamedDispatcher("AsyncPlayer");
            } else if (user.equals("pattern")) {
                log.debug("Invite for: " + user);
                dispatcher =
                getServletContext().getNamedDispatcher("PatternDetection");
            }

            if (dispatcher != null) {
                log.debug("Invite for: " + user);
                dispatcher.forward(req, null);
            }
        }
        log.debug("Exiting");
    }
}
```

In above code application dispatches a URI with user "player" or "pattern" to the appropriate application SipServlet.

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;
```

```

@ServerEndpoint("/base/{id}")
@javax.servlet.sip.annotation.SipServlet(description = "DialogicSampleDemo", name="AsyncPlayer",
applicationName="DialogicSampleDemo", loadOnStartup=0)

@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 annotation for WebSockets: *@ServerEndpoint*. This needs to be unique for each SipServlet application.
2. 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 the SIP routing. Note that the OCCAS7 method of mapping SIP messages to desired the application is accomplished by using JNDI configuration as shown in [Configuring the Platform's SIP Router](#).
 - 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.
3. 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.
 4. 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.

Mapping SIP Traffic to the Application

SIP traffic is directed by the platform to the appropriate application as defined by the application router. See [Configuring the Platform's SIP Router](#) section for further details.

Dialogic JSR 309 Connector Initialization

The first method of an application that is going to be invoked will be an `init()` method. In this method, the application loads the application's properties file *dlgc_sample_demo.properties*. Once the SIP container has been initialized, it will invoke each application's `servletInitialized()` method in the order defined in *sip.xml*.

In this `servletInitialized()` method, the application calls the `initDriver()` method, which obtains the Dialogic JSR 309 Connector automatic 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. 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 creates new connector properties by taking information from defines in the application properties file to be used later when initializing the connector configuration.

```
...
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:

```
${DOMAIN_HOME}/config/Dialogic/log4j2.xml
```

As per *log4j2.xml* configuration, Dialogic JSR 309 Connector and Verification Applications log output file can be found at:

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

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="true">
      <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="INFO">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="base" level="INFO">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="digit" level="INFO">
      <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 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* (for 309 connector), and *INFO* (for verification demo).

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:

```
<Jan 1, 2016 6:23:17 PM EST> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport TCP)" is listening on port 5061>
<Jan 1, 2016 6:23:17 PM EST> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport TCP)" is listening on port 5061>

18:23:19.407 [[ACTIVE] ExecuteThread: '7' for queue: 'weblogic.kernel.Default (self-tuning)'] ERROR base.ConfigProperty - LoadProperties() - IOException:
java.io.FileNotFoundException:
/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/config/Dialogic/dlgc_sample_demo.properties (No such file or directory)

java.io.FileNotFoundException:
/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/config/Dialogic/dlgc_sample_demo.properties (No such file or directory)

    at java.io.FileInputStream.open0(Native Method) ~[?:1.8.0_65]
    at java.io.FileInputStream.open(FileInputStream.java:195) ~[?:1.8.0_65]
```

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/apachetomcat8/mss-4.0.21-apache-tomcat-8.0.26
/conf/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 Verification Application in Eclipse IDE

The Dialogic JSR 309 Connector distribution comes with necessary configuration files and content needed to build Dialogic sample applications. This section is going to provide the steps on how to create, compile, build, and debug the provided Verification Application using Eclipse IDE.

Prerequisites

The following components must be installed:

- JDK 1.8.0_45

Note: Latest version of 1.8 JDK is used because this is a version supported by OCCAS 7.

- Eclipse KDE (Eclipse Standard SDK).
- Required third-party library (JAR) files:
 - *jain-sip-ri-1.2.256.jar*
 - *json_simple-1.1.jar*
 - *jsr173_1.0_api.jar*
 - *log4j-api-2.2.jar*
 - *log4j-core-2.2.jar*
 - *log4j-slf4j-impl-2.2.jar*
 - *org.osgi-3.0.0.jar*
 - *slf4j-api-1.7.5.jar*
 - *xbean.jar*

Note: All referenced above library (JAR) files are already provided as part of the project directory under *lib/project/3rdParty*. These are necessary in order to build the project correctly.

- Required platform specific library (JAR) files:
 - *javax.servlet_2.2.0.0_3-0.jar*
 - *javax.websocket_1.0.0.0.jar*
 - *sipservlet-api.jar*
 - *mscontrol.jar*

Note: All referenced above library (JAR) files are also necessary in order to build the project correctly. These JAR files are not provided with a distribution. These files need to be copied from OCCAS 7.0 platform into the project's library folder under: *lib/project/AS*.

Creating the Build Environment

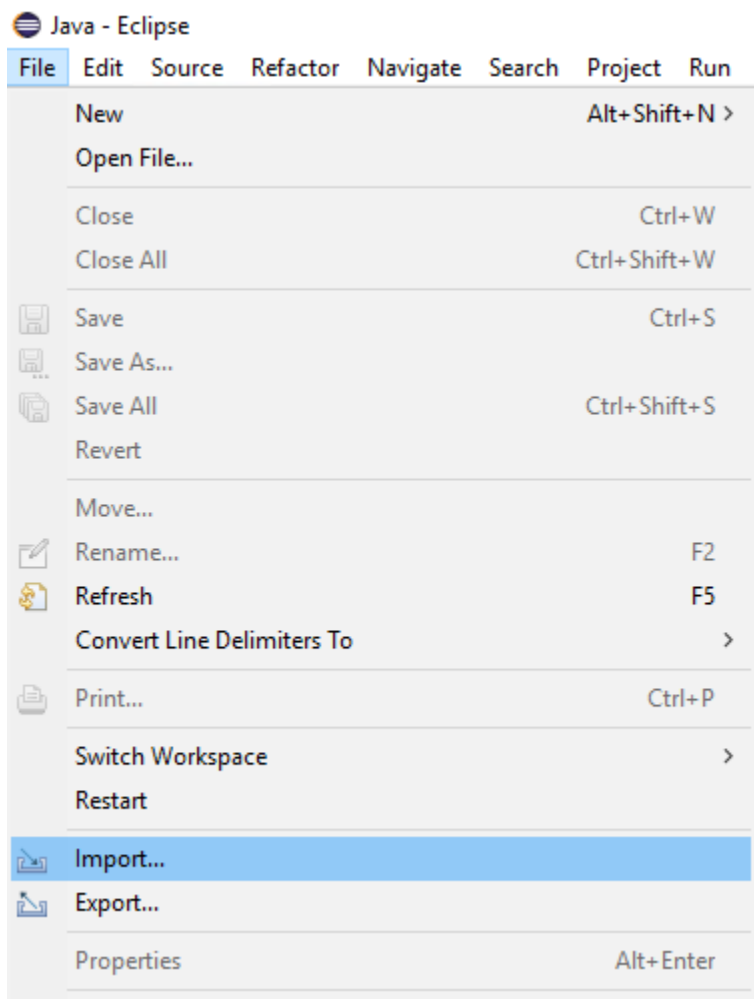
To create a Verification Application project, follow the steps below:

1. [Importing the Project from Distribution](#)
2. [Configuring the Project](#)
3. [Building the Project](#)

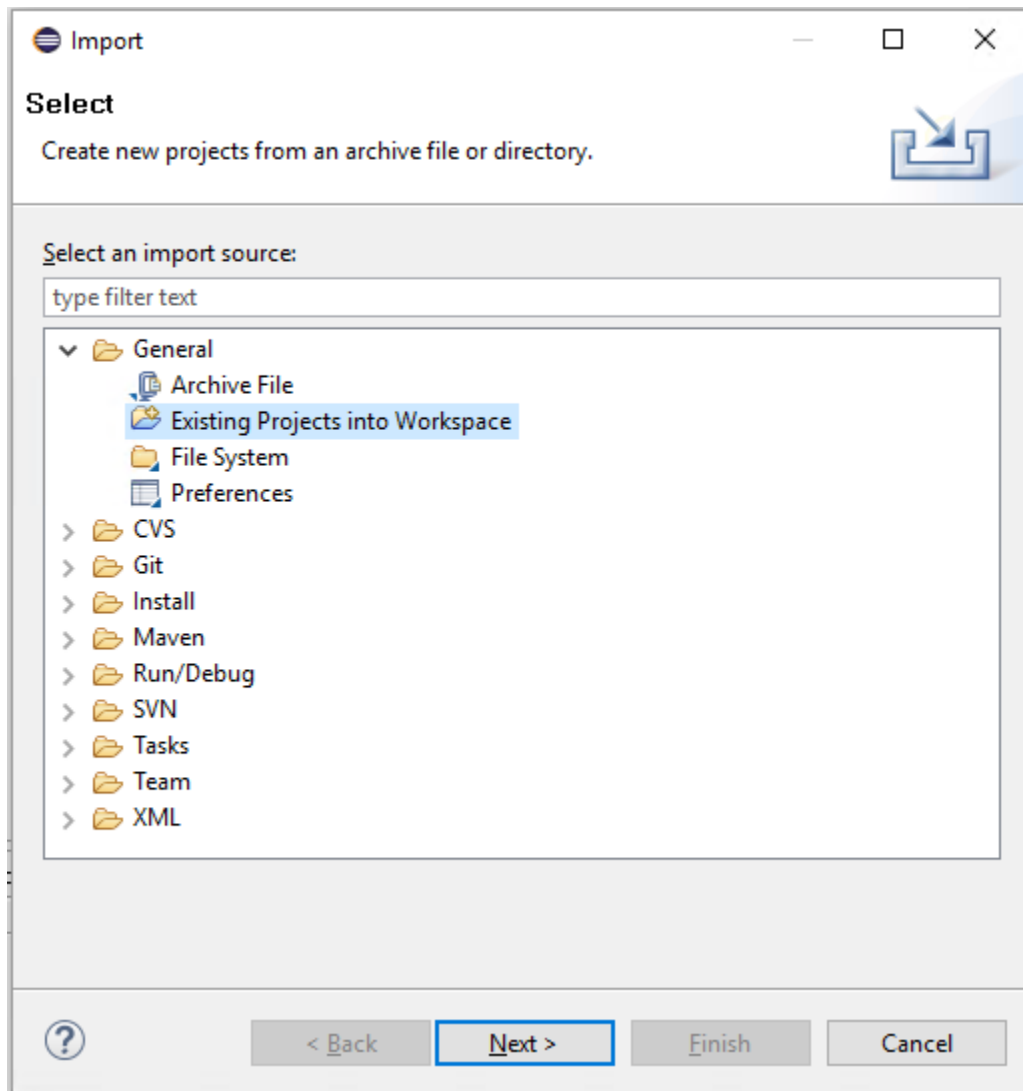
Importing the Project from Distribution

From the distribution package, unzip the *verification.zip* file from the *DlgcJSR309/application* directory and save it to a known location on your system.

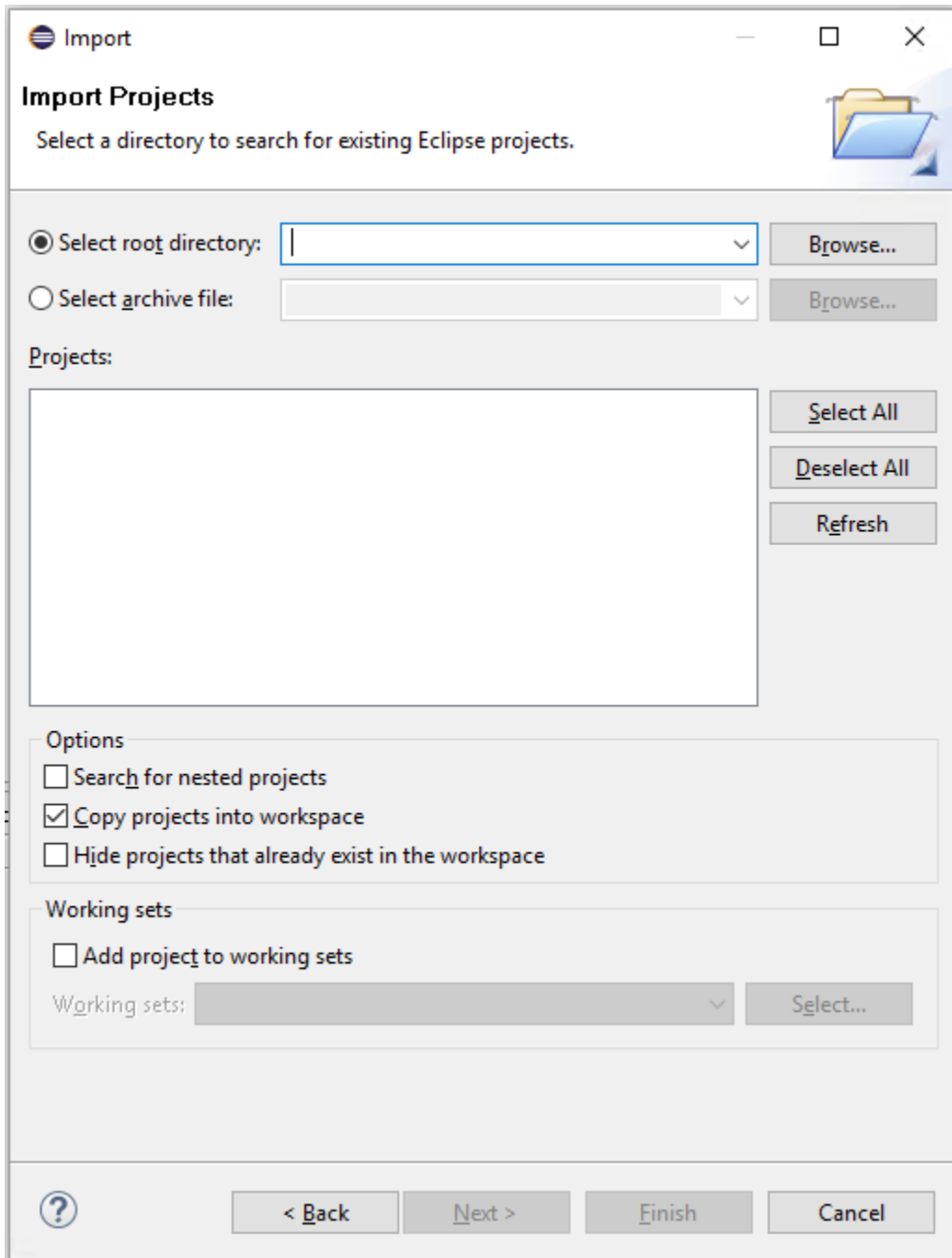
1. Copy all library (JAR) files into the newly extracted project *lib* directory as described in [Prerequisites](#).
2. Open **Eclipse IDE** and click **File > Import**:



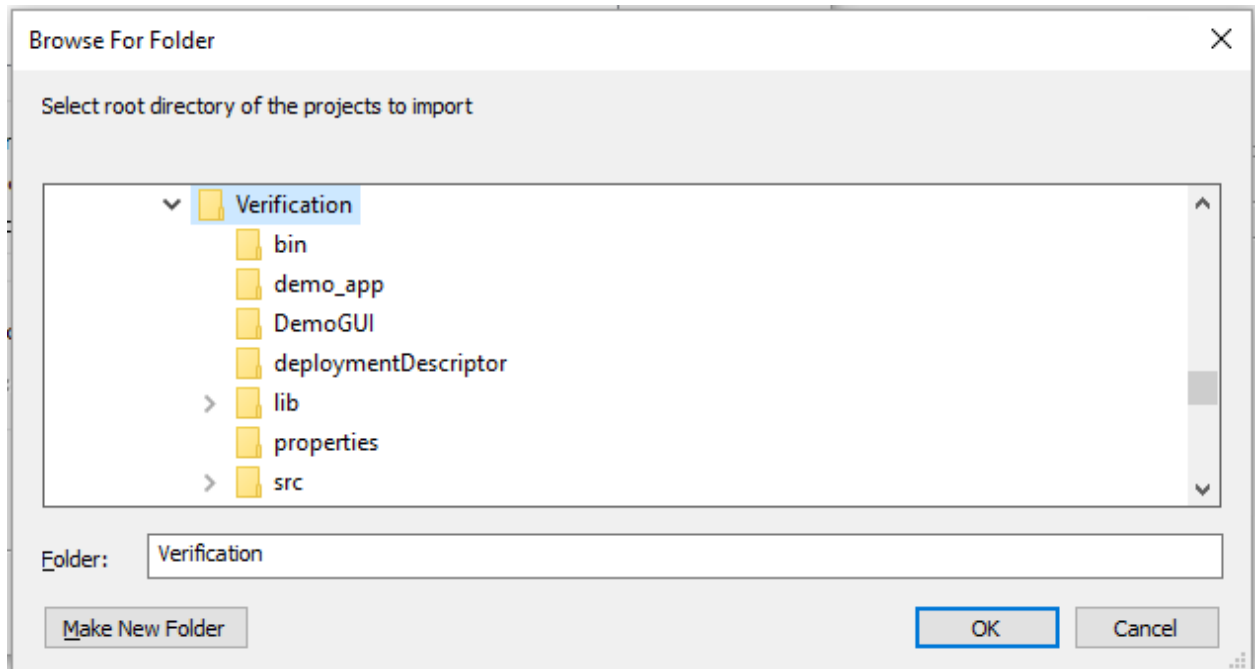
3. In the **Import** window, select **Existing Projects into Workspace** option, and then click **Next**.



4. In the **Select root directory** field, click **Browse**.



5. In the **Browse For Folder** window, navigate to the extracted project directory and click **Ok**.



6. In the **Import** screen, verify that **Copy project into workspace** is selected and click **Finish**.

Import

Import Projects

Select a directory to search for existing Eclipse projects.

☒ Select root directory: E:\Projects\Verification **Browse...**

☐ Select archive file: **Browse...**

Projects:

- ☒ Verification (E:\Projects\Verification)

Select All
Deselect All
Refresh

Options

- ☐ Search for nested projects
- ☒ Copy projects into workspace
- ☐ Hide projects that already exist in the workspace

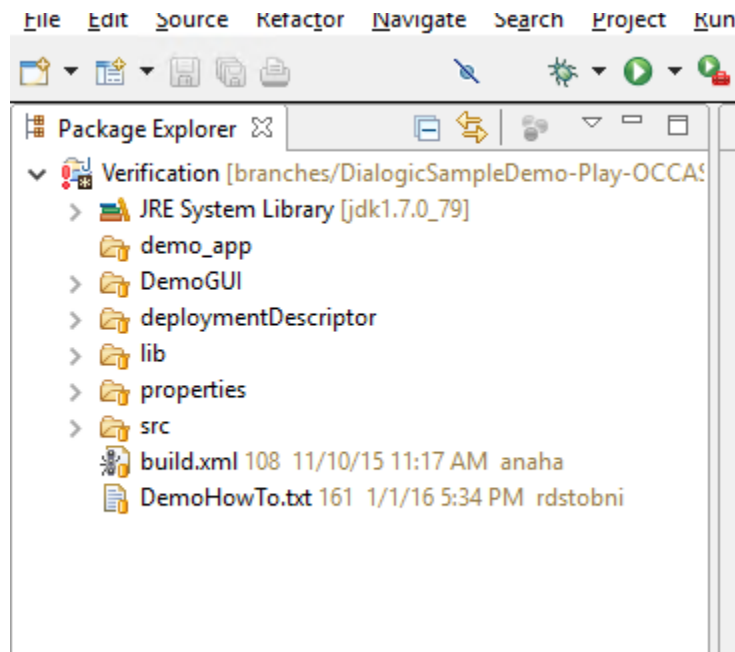
Working sets

- ☐ Add project to working sets

Working sets: **Select...**

Finish

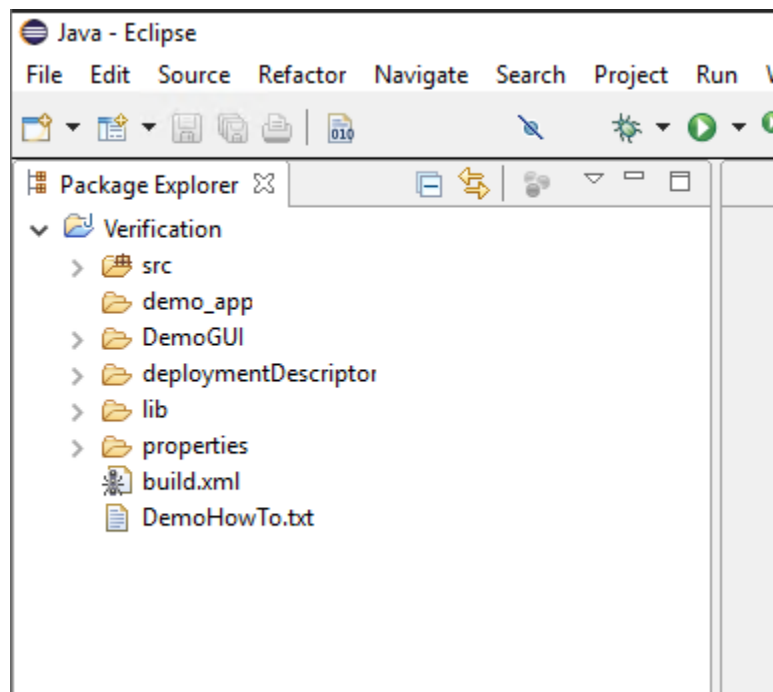
This will now create a project and its content will be shown under **Package Explorer** of the KDE.



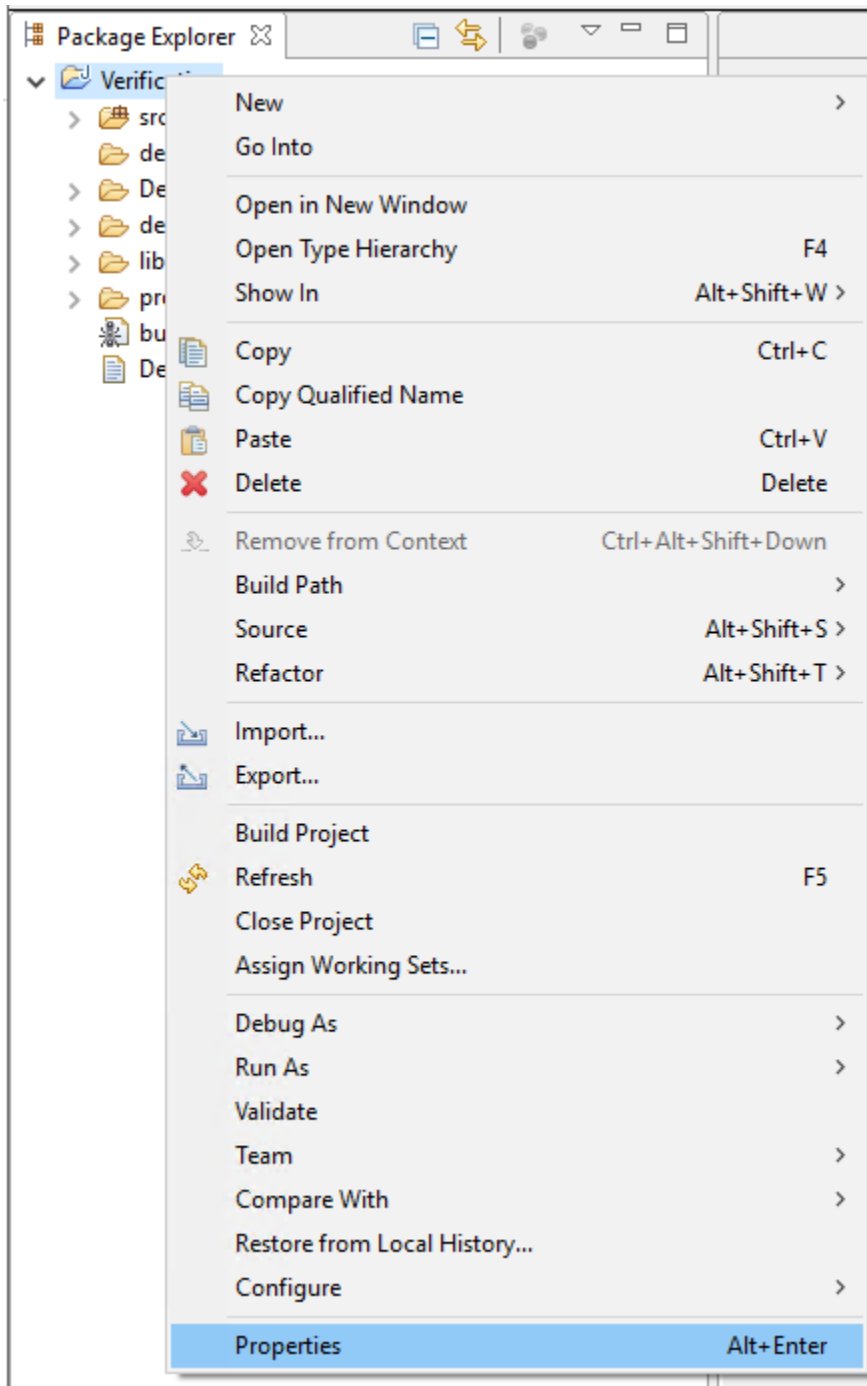
Configuring the Project

Configure the project as follows:

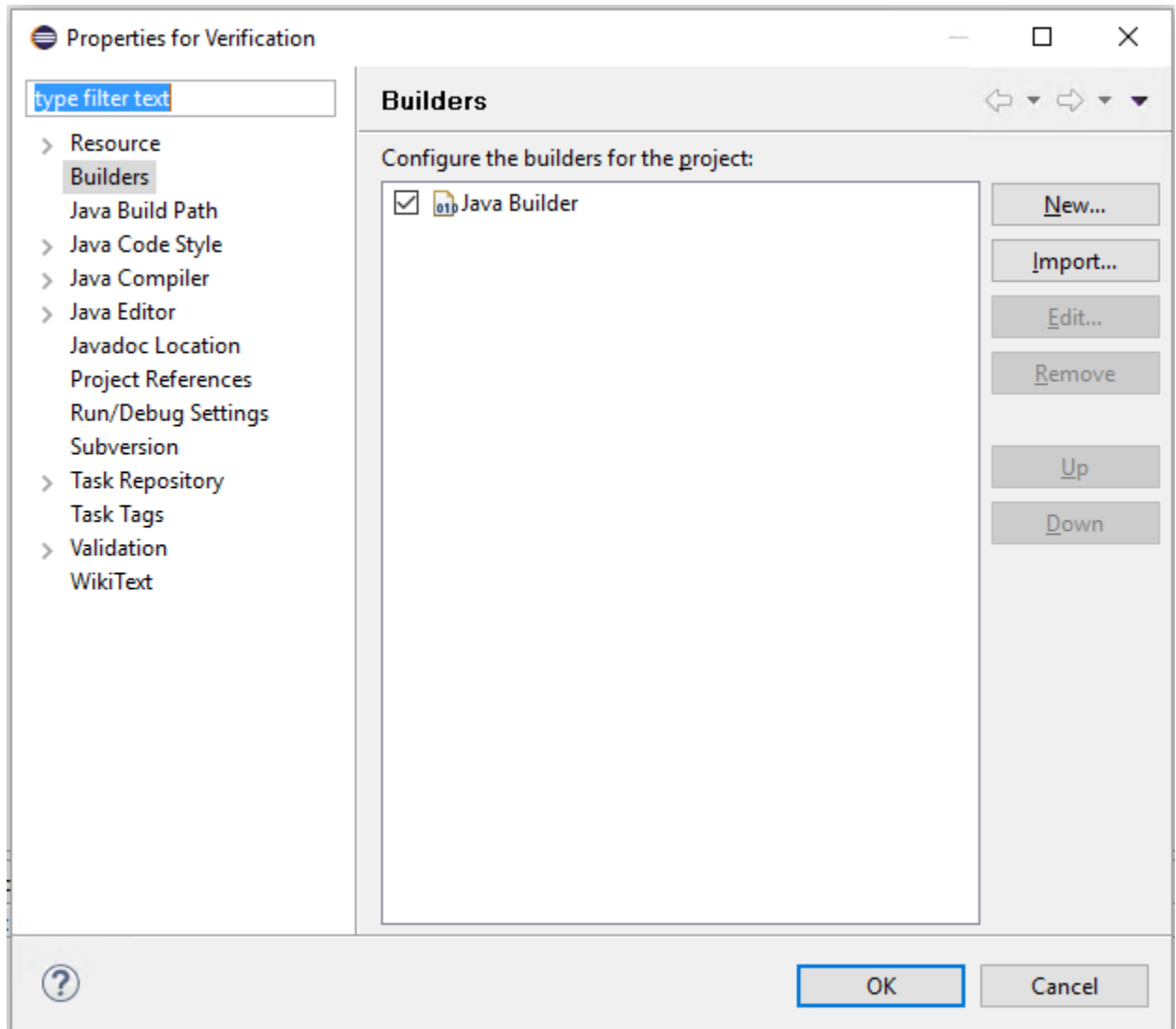
1. Expand the newly imported project in Eclipse as shown below.



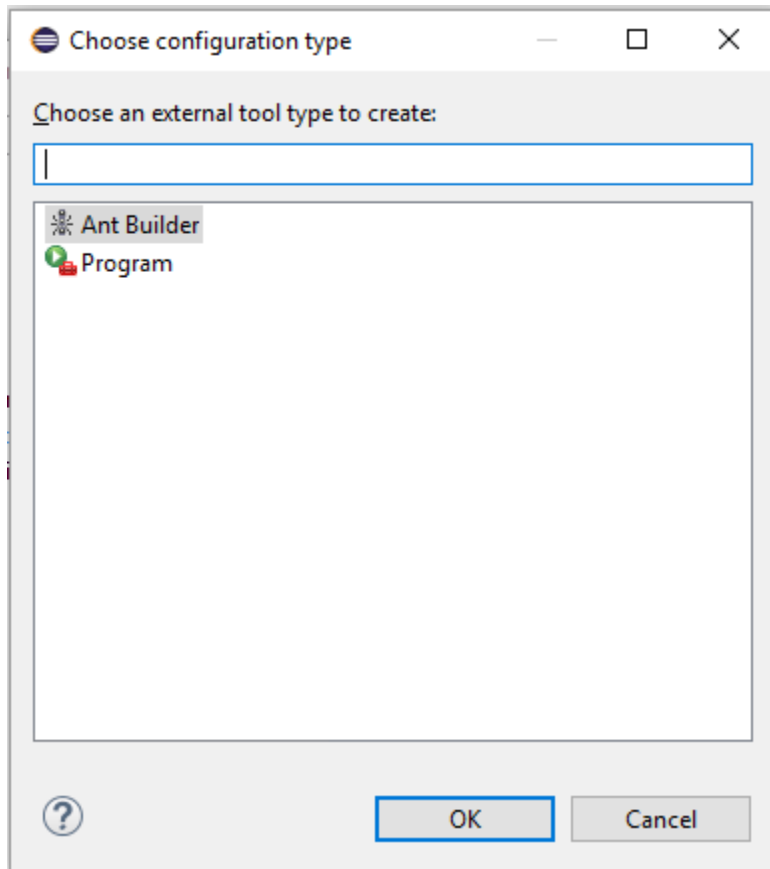
2. Select the project, right click it, and then click **Properties**.



3. In the **Properties for Verification** configuration window, click **Builders** and then click **New**.



4. To begin configuring the project to use ANT builder instead of Java Builder, select **Ant Builder** in the **Choose configuration type** window and click **OK**.



The following **Edit Configuration** window appears.

Edit Configuration

Edit launch configuration properties

Please specify the location of the external tool you would like to configure.

Name:

Main Refresh Targets Classpath Properties JRE Environment Build Options

Buildfile:

Browse Workspace... Browse File System... Variables...

Base Directory:

Browse Workspace... Browse File System... Variables...

Arguments:

Variables...

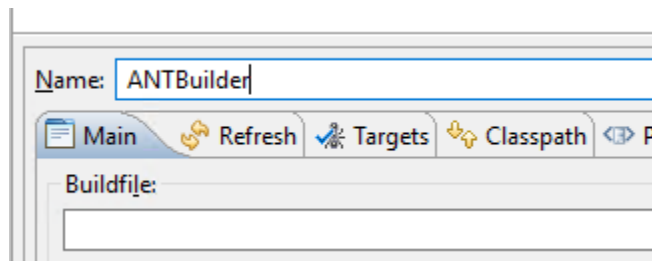
Note: Enclose an argument containing spaces using double-quotes (").

☒ Set an Input handler

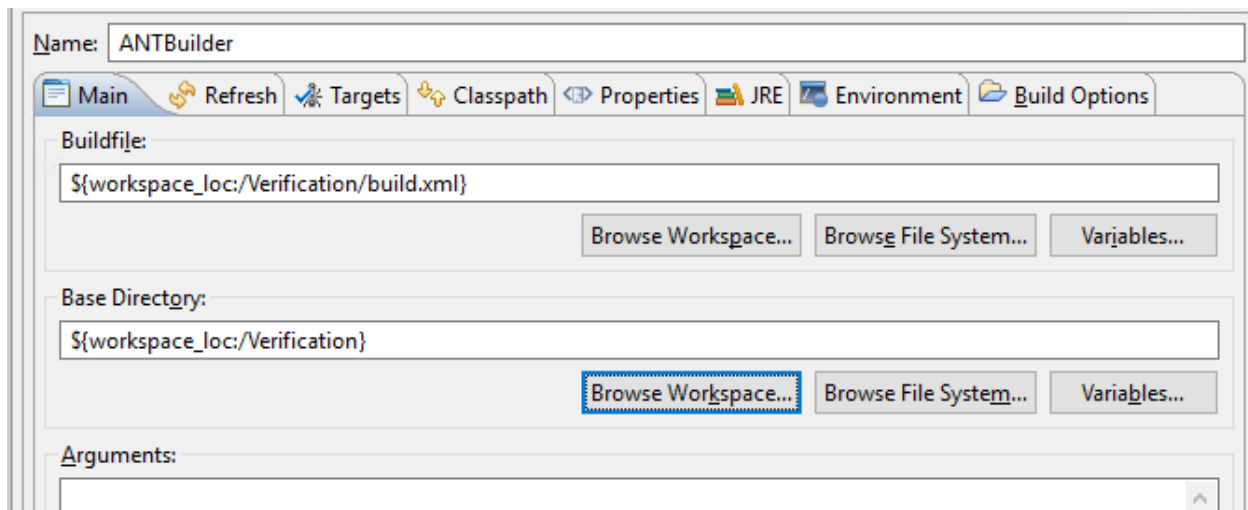
Apply Revert

? OK Cancel

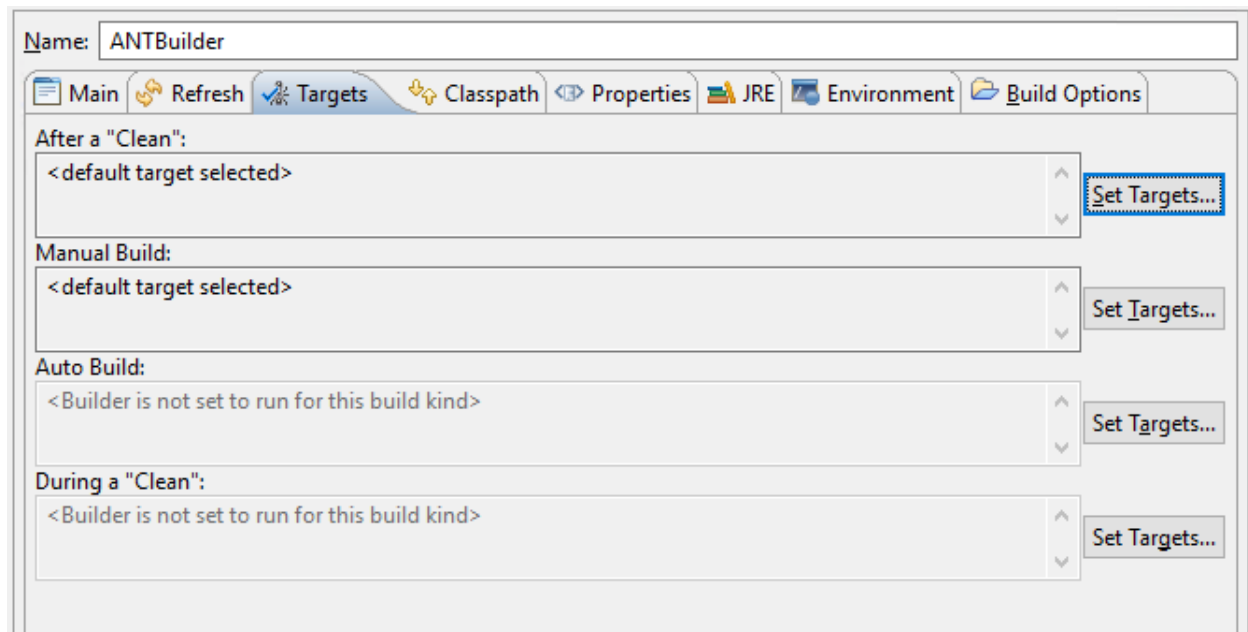
5. In the **Edit Configuration** window, configure the properties as follows:
- Fill in the name of the builder in the **Name** field for clarity (for example, *ANTBuilder*).



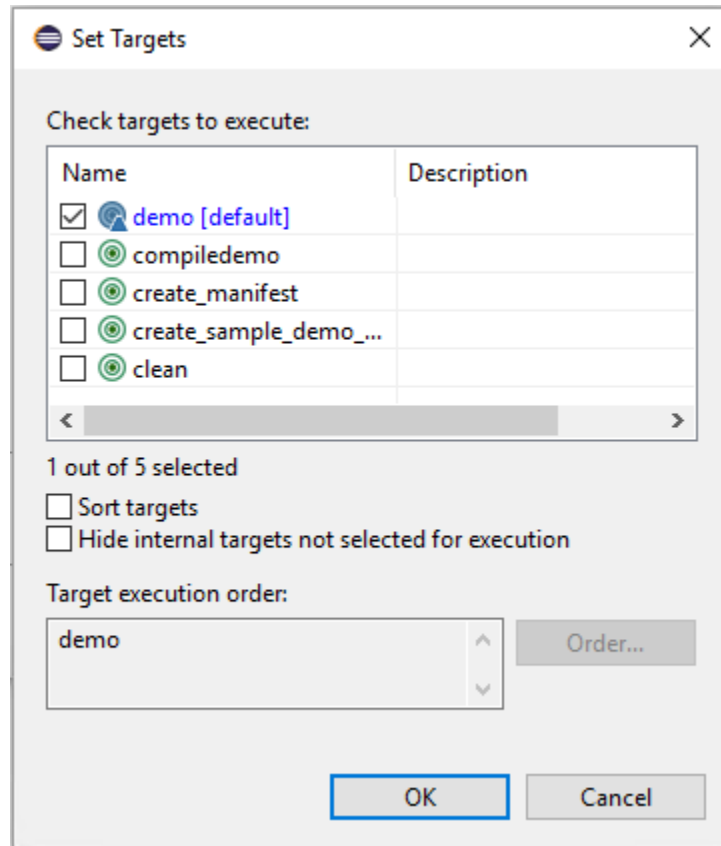
- On the **Main** page, the **Buildfile** and **Base Directory** fields must be filled in. **Buildfile** should point to the *build.xml* file under a project directory and the **Base Directory** should point to the base directory of the project.



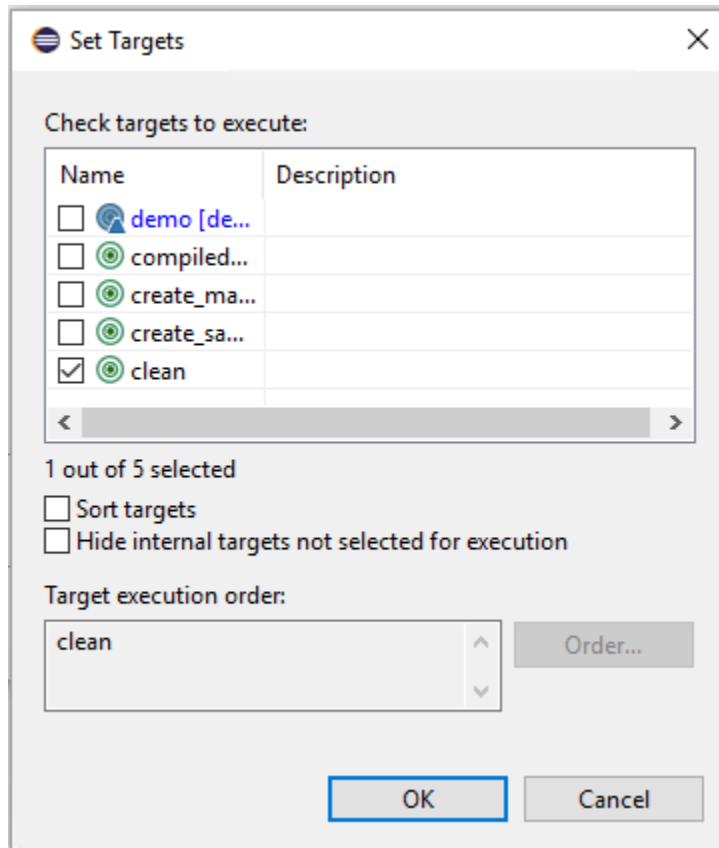
- Click the **Targets** tab.



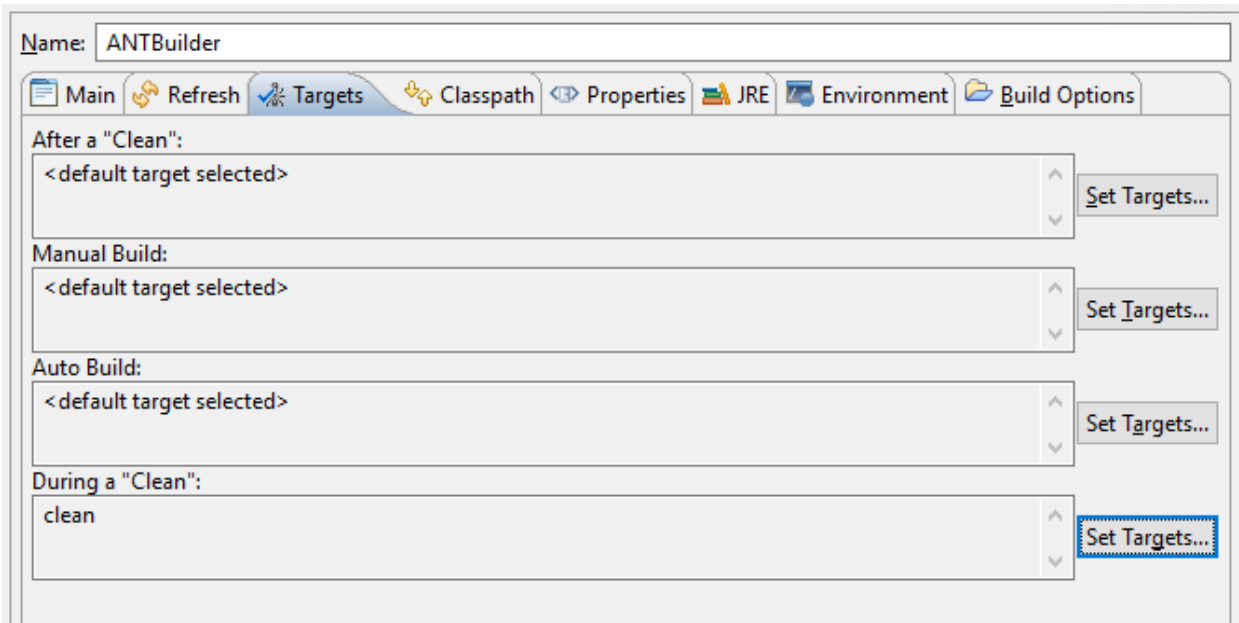
- d. Configure the **Auto Build, After a "Clean"**, and **Manual Build** fields. To do so, click **Set Targets** and select **demo [default]** for each field, and then click **OK**. An example of the **Set Targets** window is shown below.



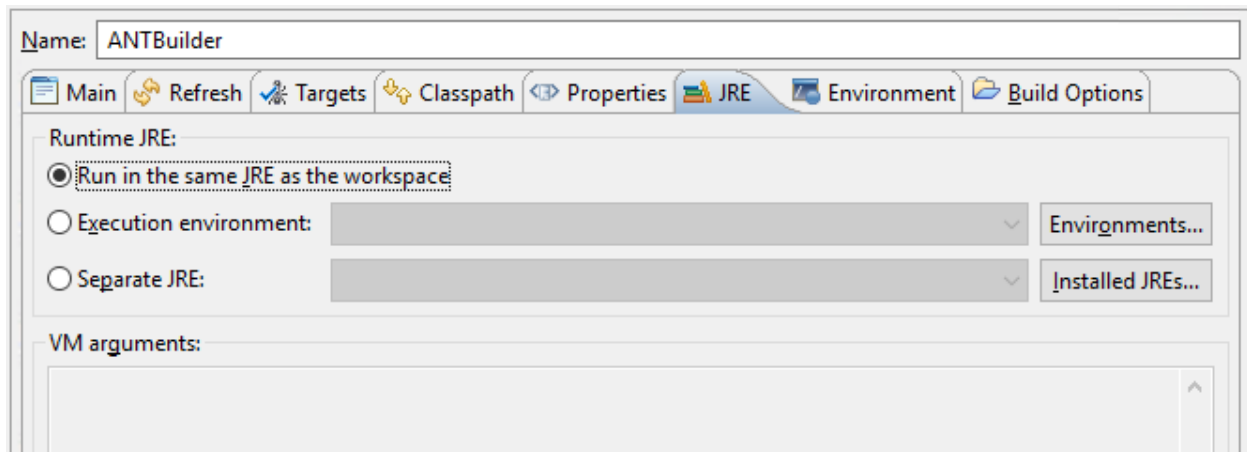
- e. To configure the **During a "Clean"** field, click **Set Targets** and select **clean**, as shown below, and then click **OK**.



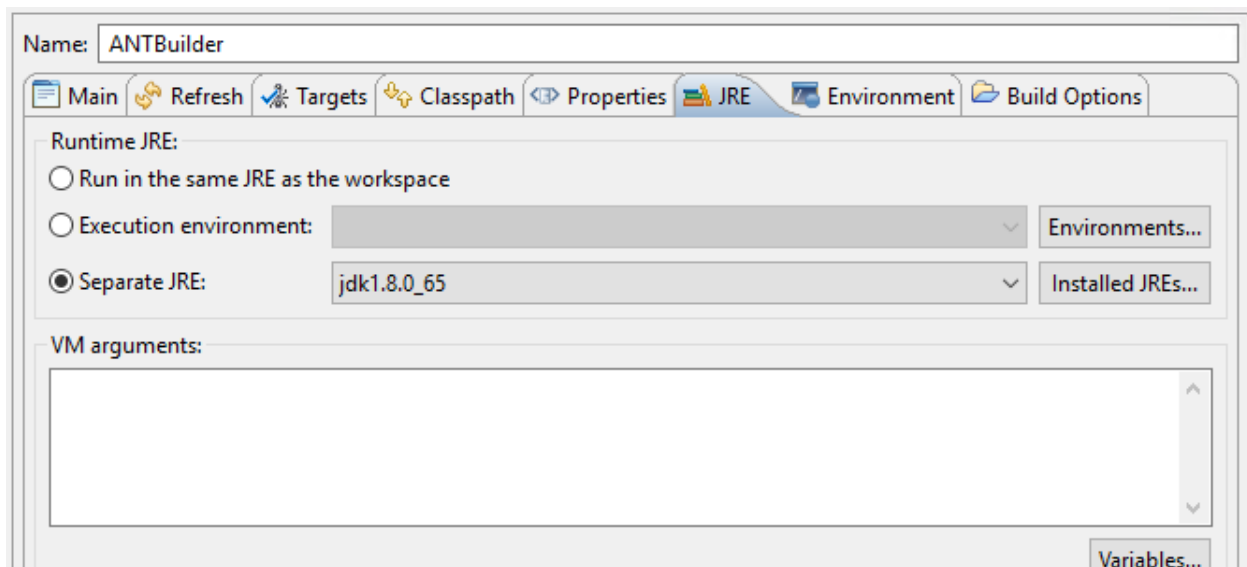
Once configured, the **Targets** page will look like this.



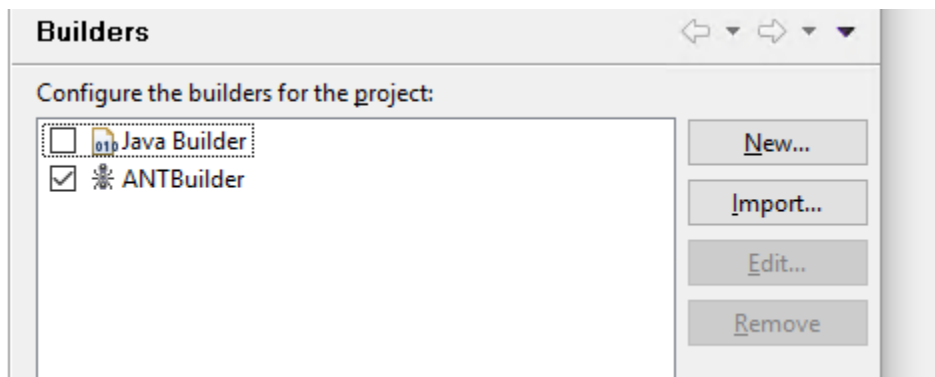
- f. Click the **JRE** tab.



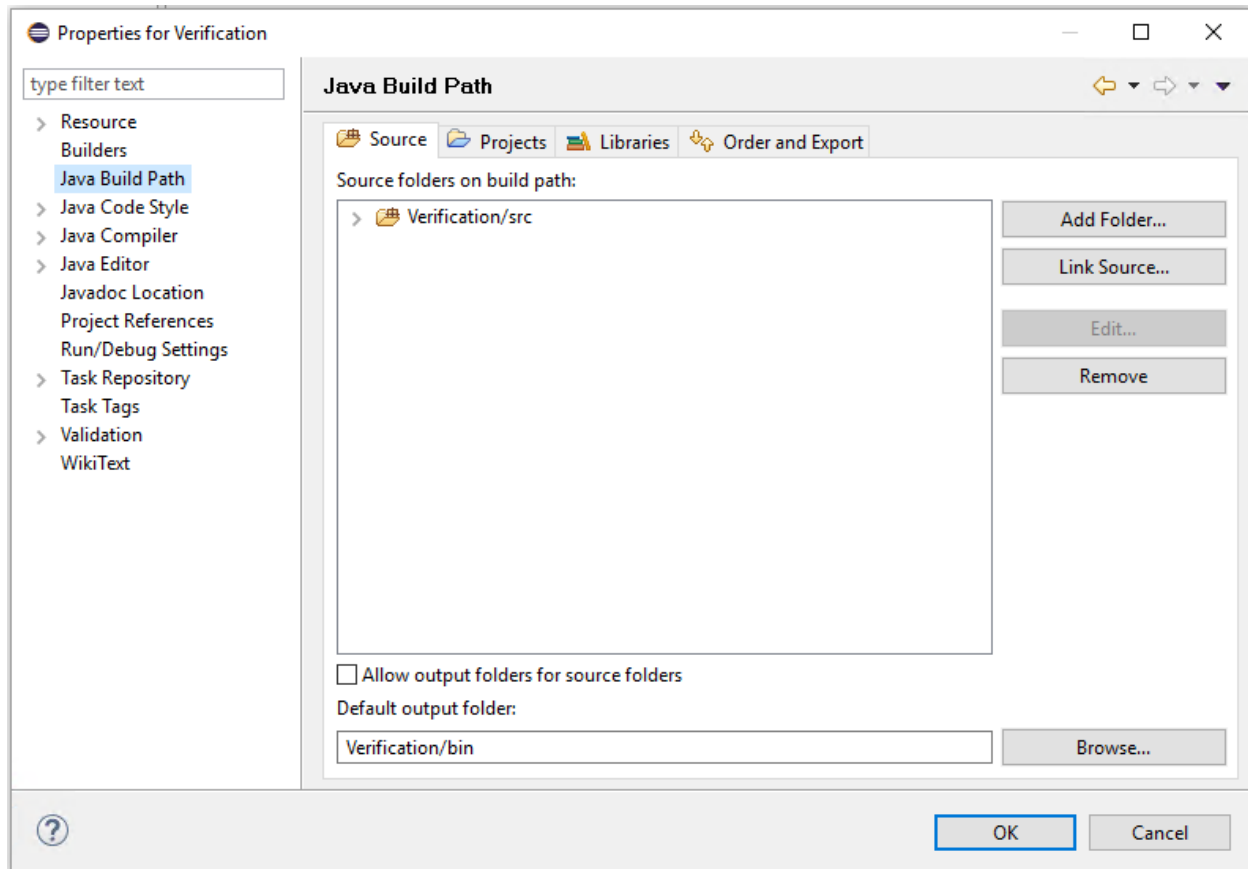
- g. On the **JRE** page, specify the correct JDK in the **Separate JRE** field, as shown below, click **Apply**, and then click **OK**.



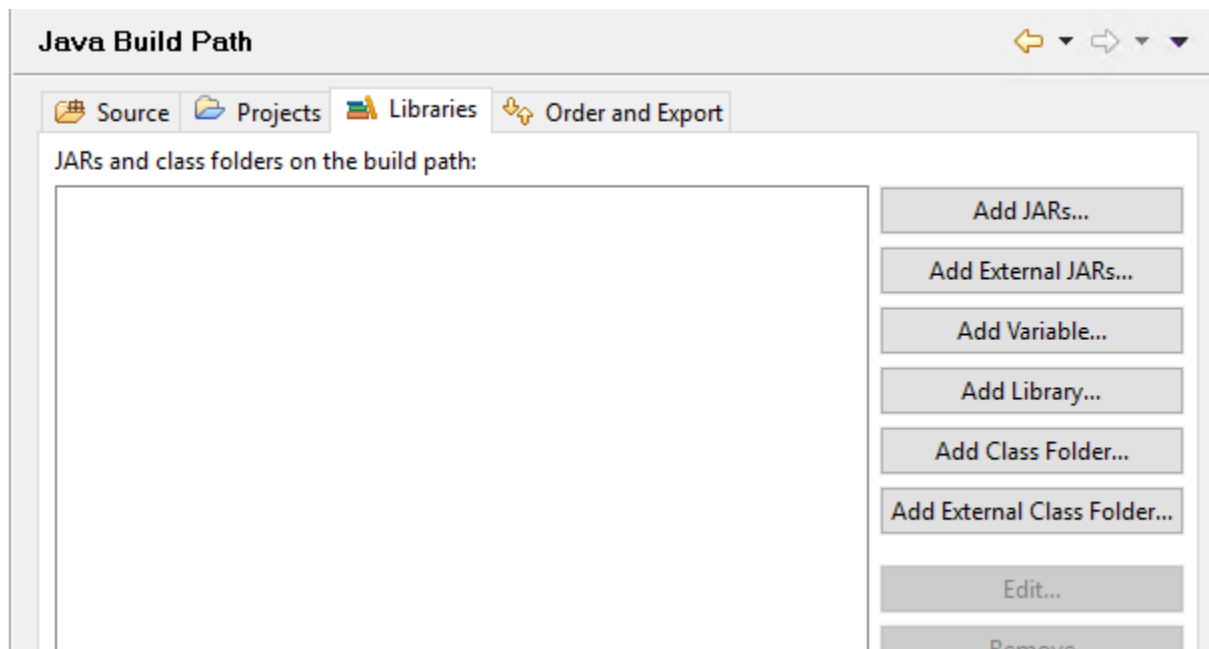
6. In the **Properties for Verification** window, de-select **Java Builder** and leave **ANTBuilder** selected.



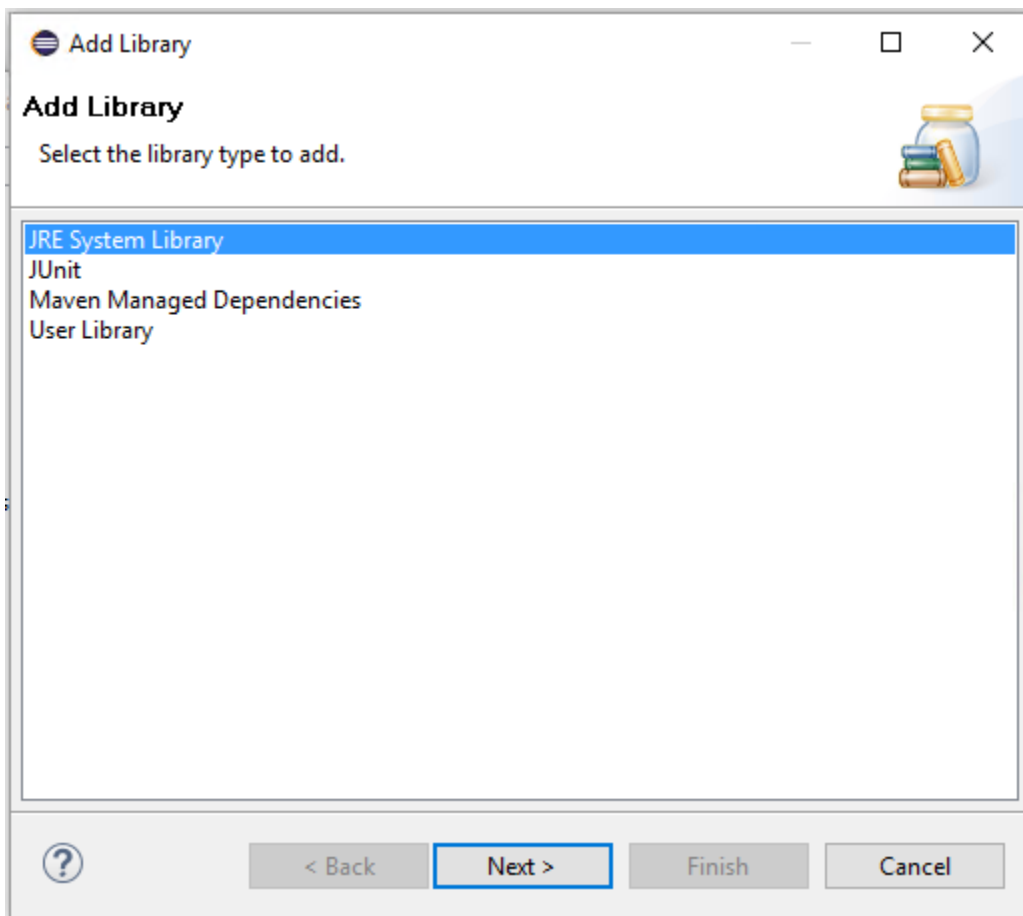
7. Select **Java Build Path** to configure it.



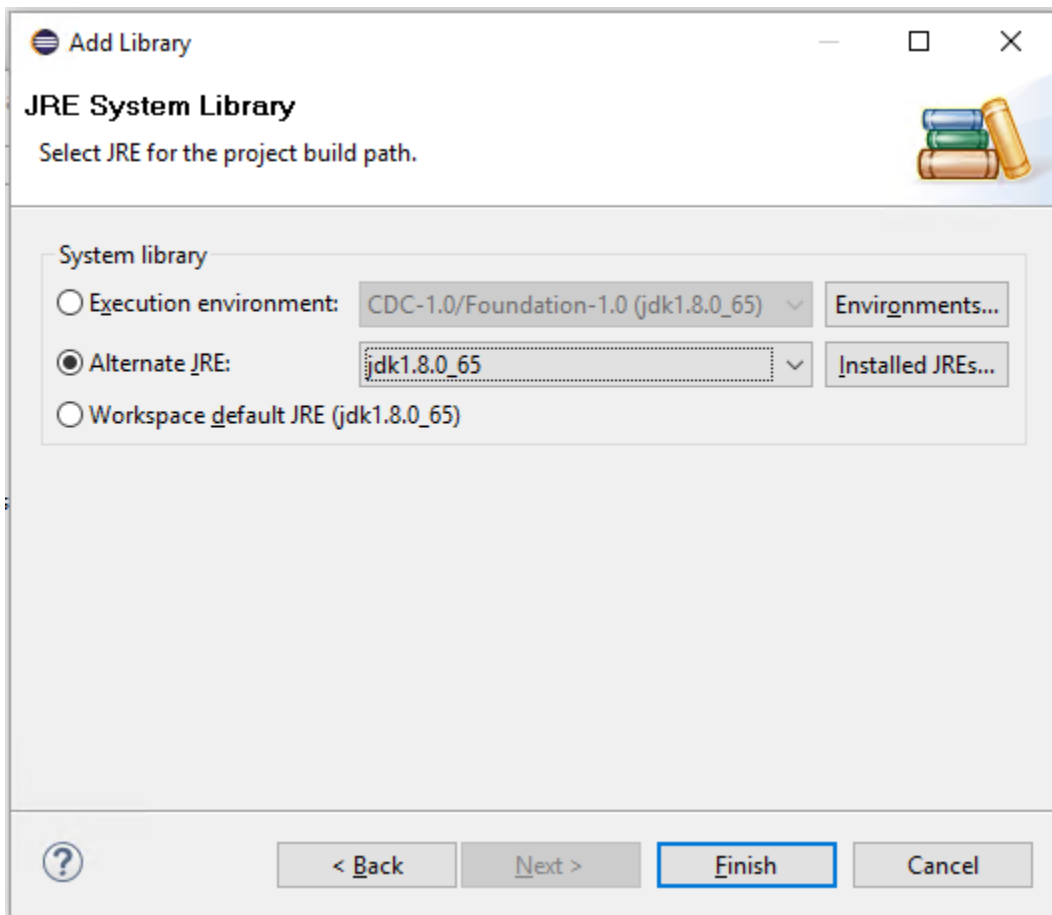
8. Click the **Libraries** tab and then click **Add Library** to begin configuring the libraries that will be used by the project. The appropriate Java library folder, the third-party JAR files, and the platform-specific files will need to be chosen.



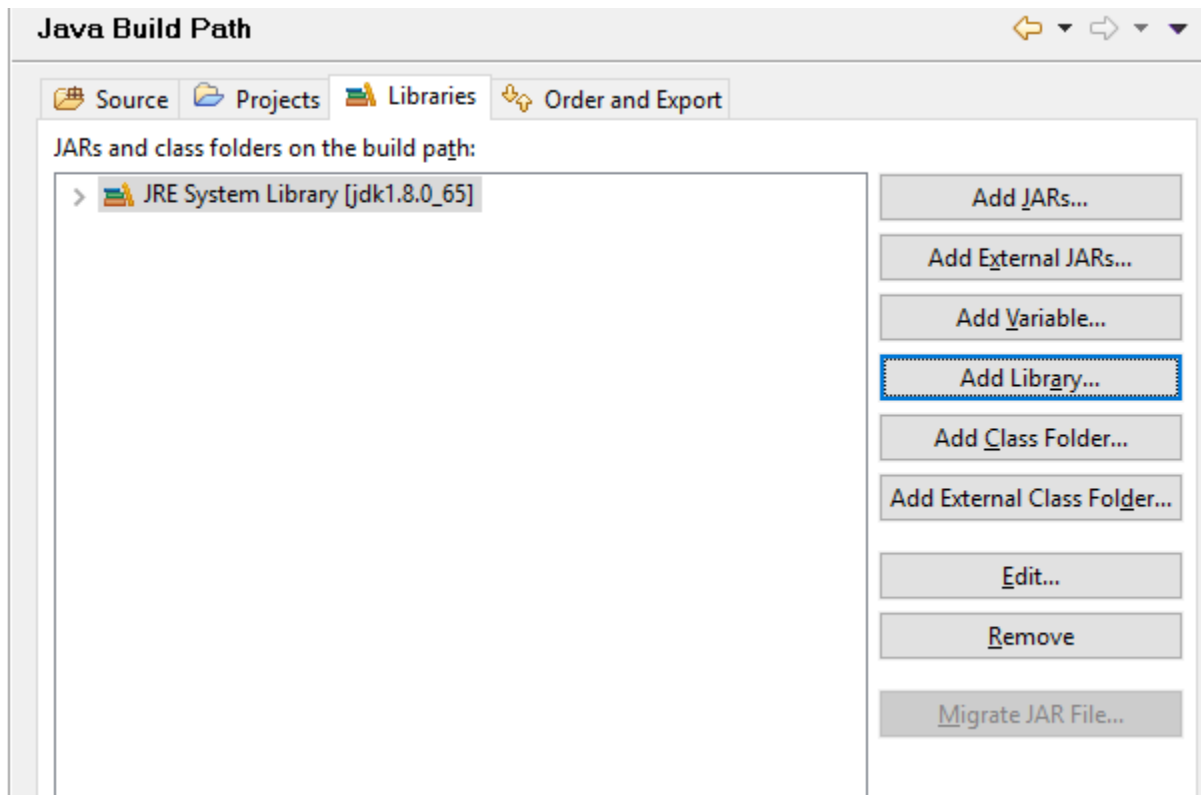
9. While **JRE System Library** is selected, click **Next**.



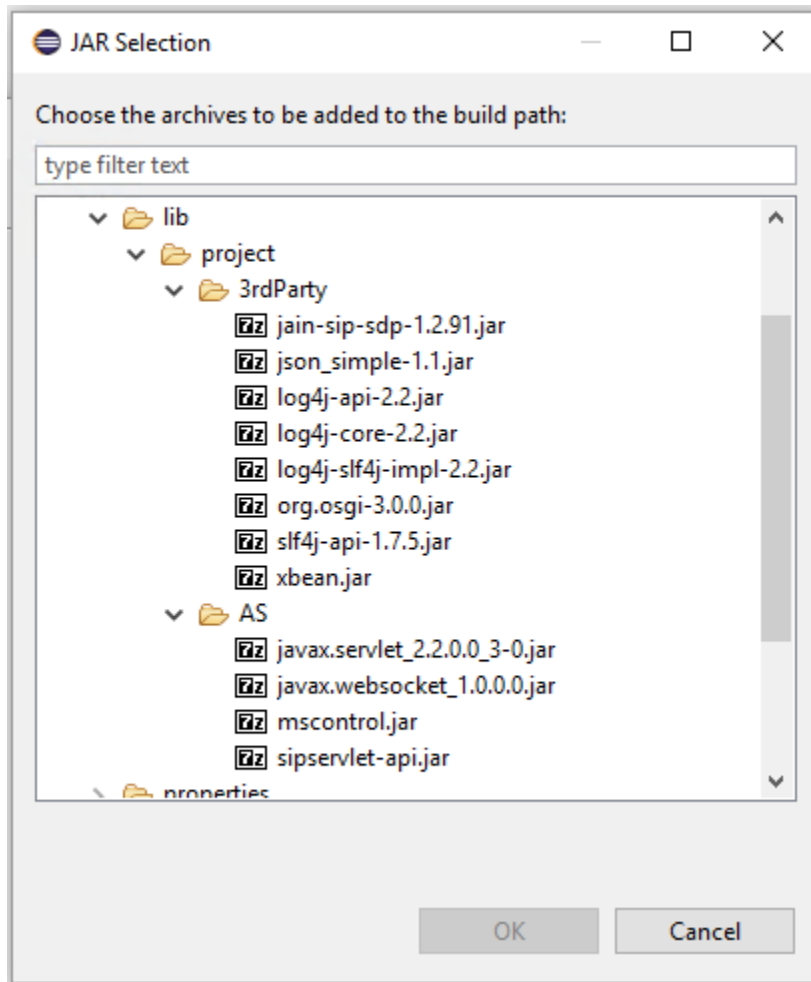
10. Verify that **Alternate JRE** points to the desired version of Java, and then click **Finish**.



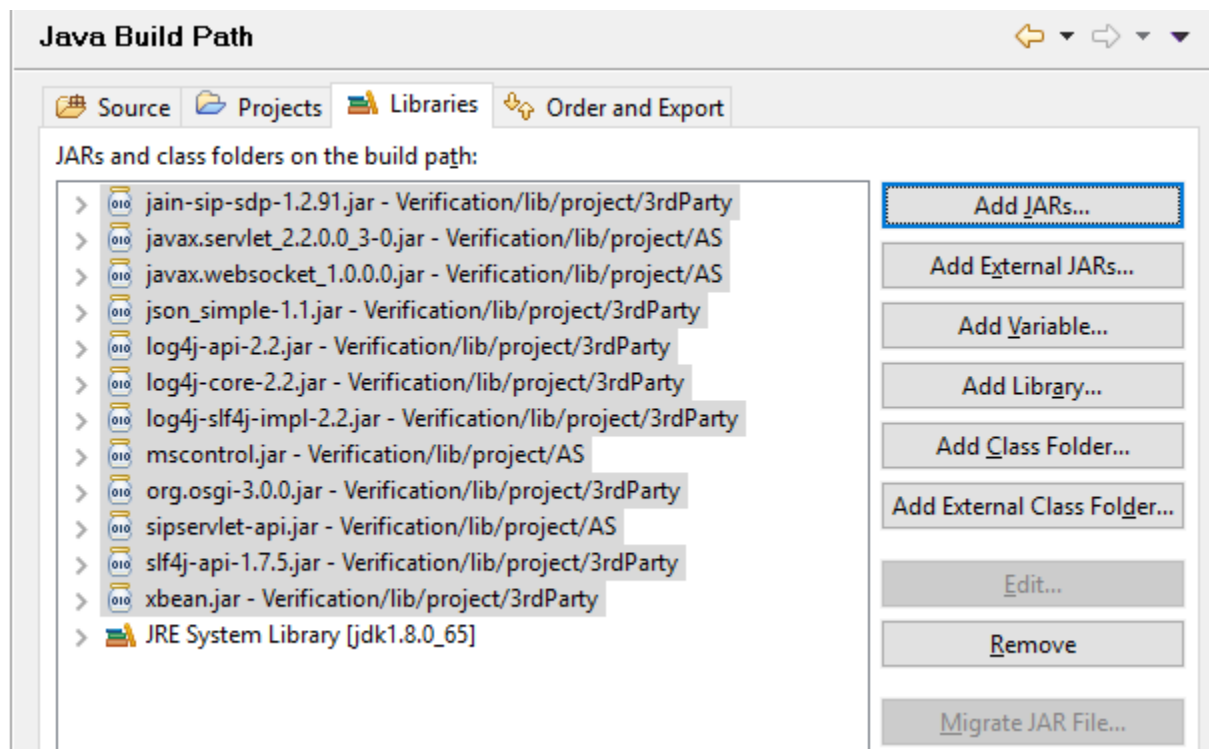
11. To specify project-dependent library (JAR) files, click **Add JARs**.



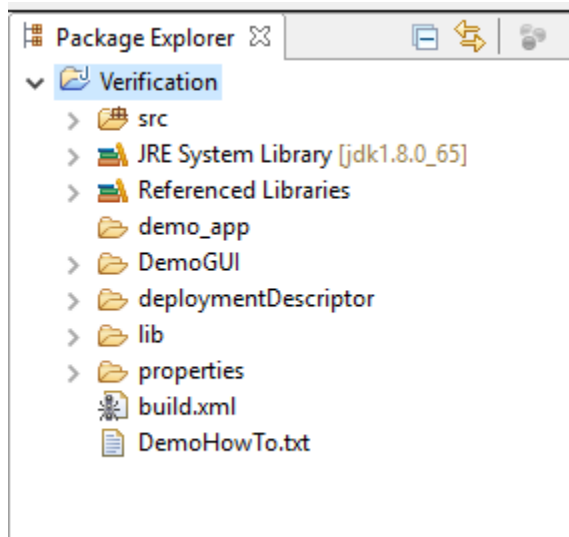
12. Select all required JAR files listed in the **3rdParty** and **AS** folders, and then click **OK**.



13. In the **Java Build Path** window, click **OK**.

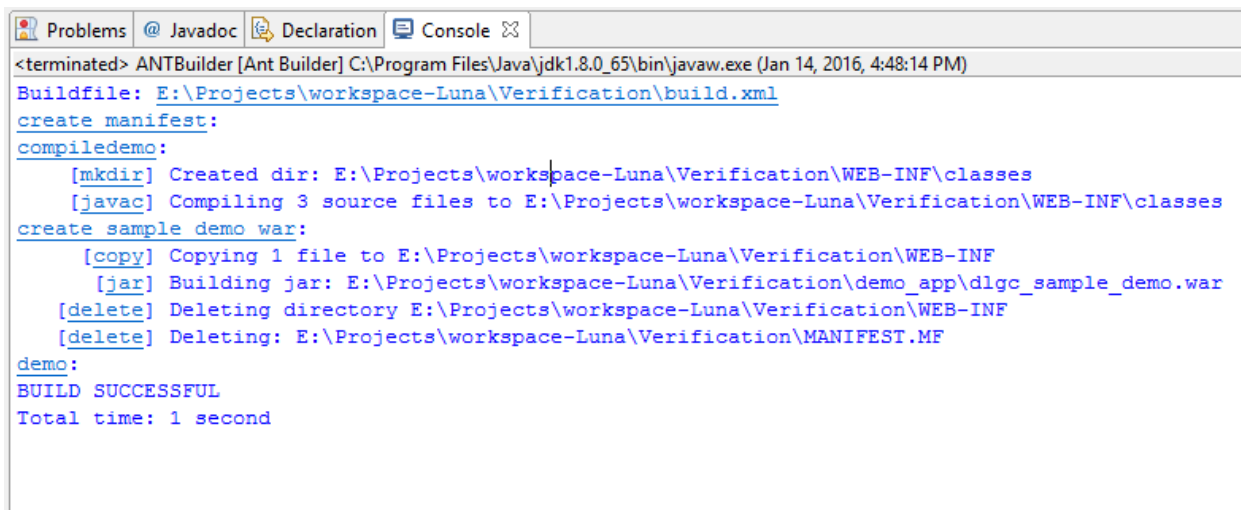


The project is configured and ready to be built.



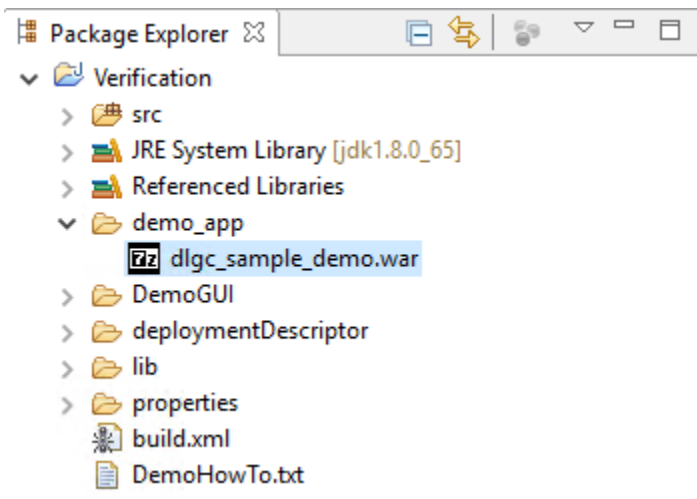
Building the Project

After a successful project installation and configuration, a project can be built. In Eclipse, select the newly created project, then go under the **Project** menu and click on **Build Project**. Successful build content will be shown in the **Console** view in Eclipse:



```
<terminated> ANTBuilder [Ant Builder] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Jan 14, 2016, 4:48:14 PM)
Buildfile: E:\Projects\workspace-Luna\Verification\build.xml
create_manifest:
compiledemo:
  [mkdir] Created dir: E:\Projects\workspace-Luna\Verification\WEB-INF\classes
  [javac] Compiling 3 source files to E:\Projects\workspace-Luna\Verification\WEB-INF\classes
create sample demo war:
  [copy] Copying 1 file to E:\Projects\workspace-Luna\Verification\WEB-INF
  [jar] Building jar: E:\Projects\workspace-Luna\Verification\demo_app\dlgc_sample_demo.war
  [delete] Deleting directory E:\Projects\workspace-Luna\Verification\WEB-INF
  [delete] Deleting: E:\Projects\workspace-Luna\Verification\MANIFEST.MF
demo:
BUILD SUCCESSFUL
Total time: 1 second
```

The newly built application WAR file will be located in the *Verification\demo_app* directory as illustrated below.



Application deployment details can be found in [Dialogic JSR 309 Verification Application](#).

Configuring Eclipse Project and OCCAS Deployed Application for Remote Debugging

In order to connect the newly created project to the deployed WAR file in OCCAS 7.0 for debugging purposes, developers need to do the following:

1. Have Eclipse successfully build the Dialogic JSR 309 Verification Application WAR file and deploy it in OCCAS 7.0.
2. Configure OCCAS 7.0 for remote debugging:
 - a. Stop OCCAS 7.0.
 - b. In OCCAS 7.0, edit the *startWeblogic.sh* script file and add the following line in the existing "Dialogic additions" section to enable remote debugging as illustrated below.

```
SAVE_JAVA_OPTIONS="${JAVA_OPTIONS}"

# Dialogic additions

LOG4J_OPTIONS="-
Dlog4j.configurationFile=${DOMAIN_HOME}/config/Dialogic/log4j2.xml"
CLASSPATH="${DOMAIN_HOME}/lib/log4j-api-2.2.jar:${DOMAIN_HOME}/lib/log4j-slf4j-impl-2.2.jar:${DOMAIN_HOME}/lib/log4j-core-2.2.jar:${CLASSPATH}"
SERIALIZATION="-Dwlss.local.serialization=false"

# Remote Debugging
DEBUG_OPTS="-Xdebug -Xrunjdwp:transport=dt_socket,address=8787,server=y,suspend=n"
# END Remote Debugging

DLGC_OPTS="${DEBUG_OPTS} ${SERIALIZATION} ${LOG4J_OPTIONS}"
# END Dialogic additions

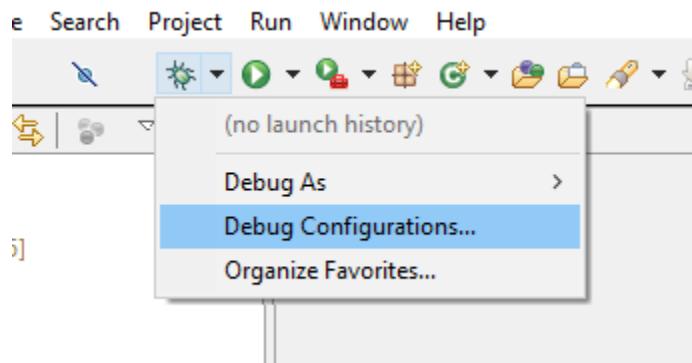
SAVE_CLASSPATH="${CLASSPATH}"
... .
```

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.

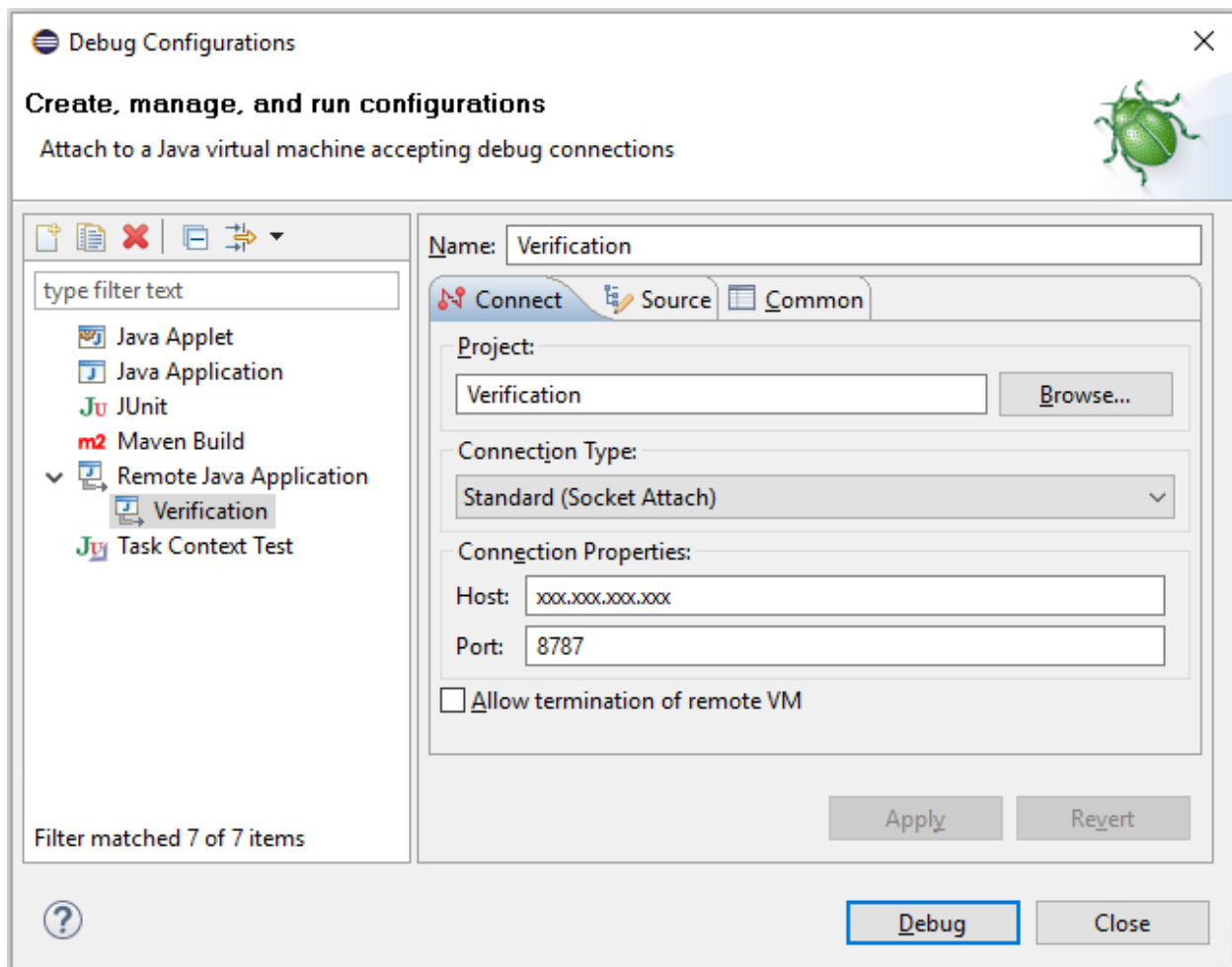
3. Start OCCAS 7 and make sure there are no errors in the console.

Eclipse Project Remote Debugging Configuration

When in Eclipse with an active Dialogic JSR 309 Connector demo project (as described in this section), the remote debugging section needs to be configured in order to set up remote debugging. In Eclipse, right click the debug icon and select **Debug Configurations**.

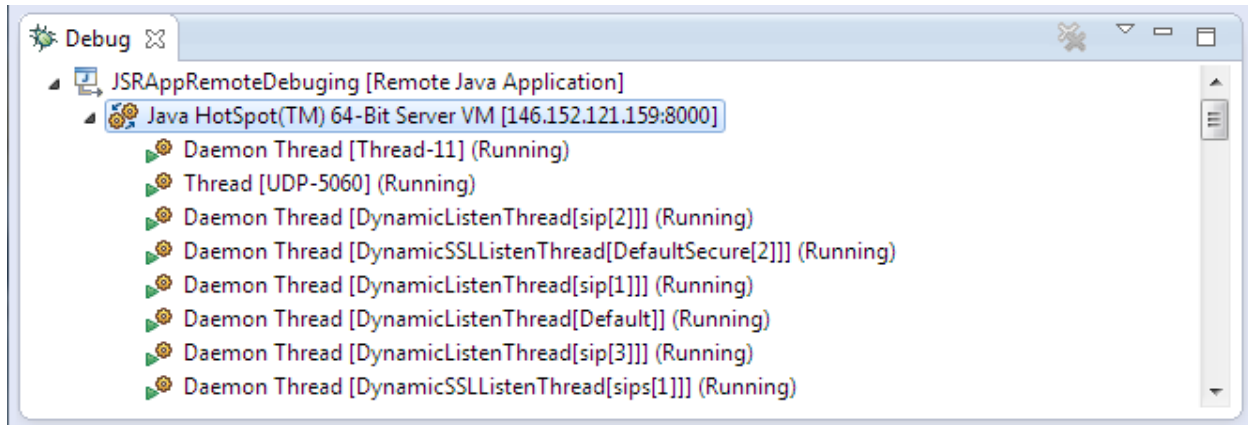


In this section double-click **Remote Java Application**. This will create a remote configuration of the existing project. Fill in the **Host** field with the appropriate IP address that points to your application server platform and the **Port** field with the debug port that was configured earlier. In this example, the debug port is 8787.



Once properly configured click **Apply**, and then click **Debug**. A successful debugging connection to the application platform will be indicated by the following content in the **Debug** window in Eclipse.

Once the **remote debugging configuration** is selected and a connection is established, the content of the **Debug** window should show running threads.



Now, the Eclipse project is connected to the build application that is deployed in OCCAS 7.0.

7. Appendix A: Dialogic JSR 309 Connector Environment Setup

About

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 OCCAS 7, but it will help build an OCCAS 7 system that can be used for verification purposes.

Steps to complete on the operating server level include the following:

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

Note: The ports that are required to be enabled in the firewall include SIP, TCP, and UDP ports 5060 and 5061 as well as 7001 which will be used by OCCAS 7.0.

If you need more details on OCCAS 7.0, refer to the OCCAS 7.0 installation instructions available from www.oracle.com.

Installing and Configuring OCCAS

This section describes the installation and configuration instructions for OCCAS 7.0.

Note: If you are familiar with OCCAS 7.0 or are planning to deploy on an existing OCCAS 7.0 setup, proceed to [Installation and Configuration](#).

- [Preinstallation Setup](#)
- [OCCAS Installation](#)
- [OCCAS Configuration](#)
- [OCCAS Startup](#)
- [Firewall Configuration](#)
- [OCCAS Verification](#)

Preinstallation Setup

1. Modify the `/etc/hosts` file:

```
xxx.xxx.xxx.xxx 'hostname'
```

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

2. Run the following command at the prompt:

```
service network restart
```

3. Create a non-root user account that OCCAS7 will be installed under. Refer to the following example:

```
useradd occas7.0
```

To set the password, use the following command and follow its instructions:

```
passwd occas7.0
```

4. Download the latest JDK .rpm file from www.oracle.com and install the JDK .rpm file. For example, the following setup is based on *jdk-8u51-linux-x64.rpm* file.

```
rpm -ivh jdk-8u51-linux-x64.rpm
```

5. Modify the .bashrc file and add the following line to match the JDK install directory:

```
# .bashrc

export JAVA_HOME=/usr/java/jdk1.8.0_51

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

6. Save the .bashrc file and then execute the following command to bring the changes into effect on the system:

```
source ./bashrc
```

OCCAS Installation

Refer to Oracle's OCCAS 7 Installation Guide for any further details and additional installation options. Note that the GUI installation does require a graphical capability of the system ("X Window System") in order for the installer to be able to display the GUI.

Log in as a non-root user (for example, *occas7*) and place the OCCAS7 installation file (*occas_generic.jar*) in the non-root user home directory (for example, */home/occas7*).

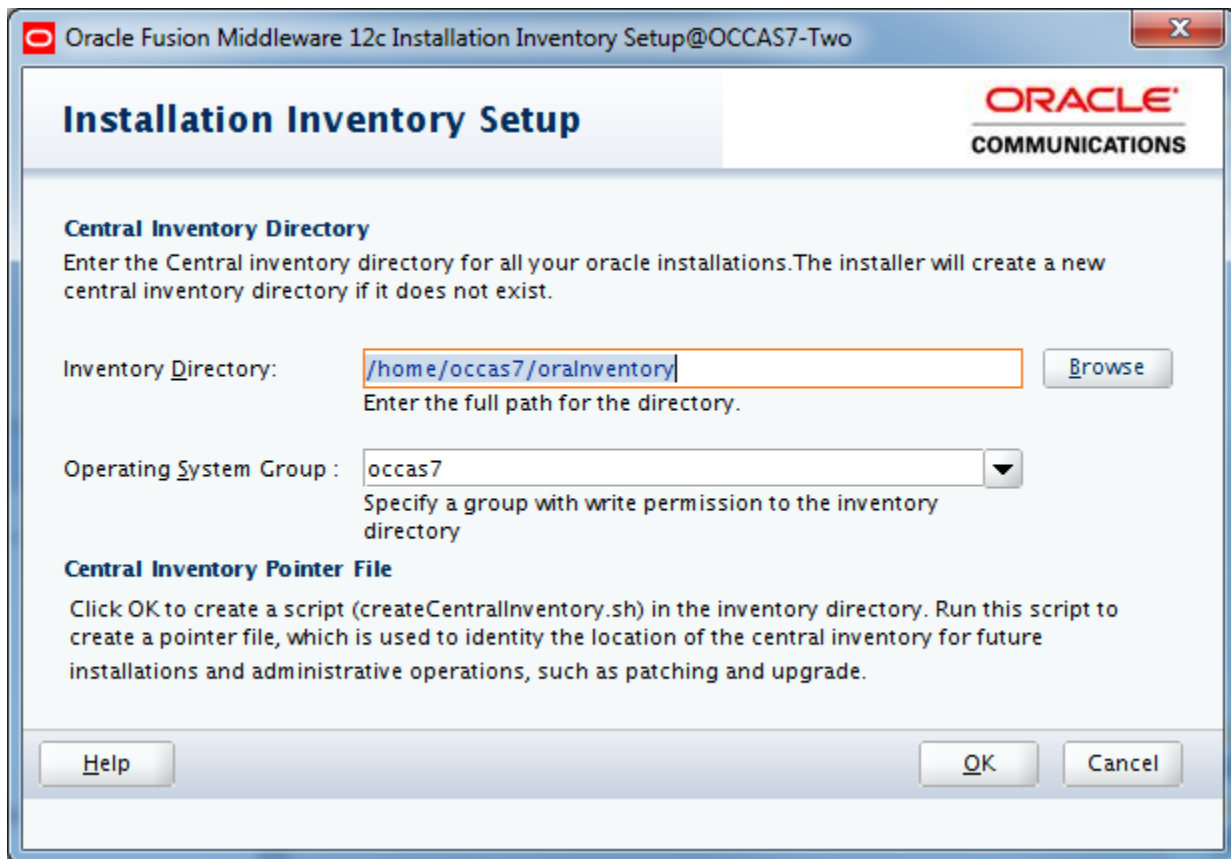
1. To install OCCAS7, execute the following command:

```
java -d64 -jar occas510_ja_generic.jar
```

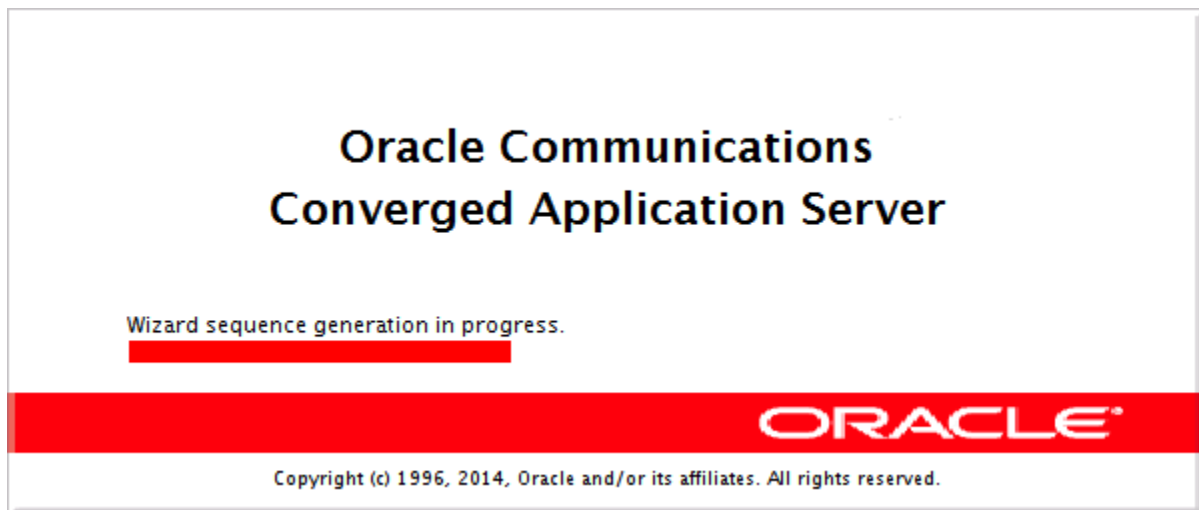
Note: You may need to change file permissions to be able to executable it.

```
occas7@OCCAS7-Two:~  
Main Options  VT Options  VT Fonts  
  
Some system prerequisite checks failed.  
You must fulfill these requirements before continuing with the installation.  
  
Continue? (yes [y] / no [n]) [n]  
n  
You have confirmed that the product cannot be installed on this platform.  
Quitting the installation.  
  
Exiting Oracle Universal Installer.  
Log(s) for this session can be found in /tmp/OraInstall2015-08-17_03-51-46PM/launcher2015-08-17_03-51-46PM.log.  
[occas7@OCCAS7-Two ~]$ exit  
logout  
Connection to 146.152.122.192 closed.  
  
WIN7LABRDS@WIN7LABRDS-PC ~  
$ ssh -Y occas7@146.152.122.192  
occas7@146.152.122.192's password:  
Warning: No xauth data; using fake authentication data for X11 forwarding.  
Last login: Mon Aug 17 14:23:27 2015 from 10.20.120.18  
/usr/bin/xauth: creating new authority file /home/occas7/.Xauthority  
[occas7@OCCAS7-Two ~]$ java -d64 -jar occas_generic.jar  
Launcher log file is /tmp/OraInstall2015-08-17_03-52-01PM/launcher2015-08-17_03-52-01PM.log.  
Extracting files.....  
Starting Oracle Universal Installer  
  
Checking if CPU speed is above 300 MHz.   Actual 2394.018 MHz   Passed  
Checking monitor: must be configured to display at least 256 colors.   Actual 16777216   Passed  
Checking if this platform requires a 64-bit JVM.   Actual 64   Passed (64-bit not required)  
Checking temp space: must be greater than 300 MB.   Actual 42235 MB   Passed  
  
Preparing to launch the Oracle Universal Installer from /tmp/OraInstall2015-08-17_03-52-01PM  
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was removed in 8.0  
█
```

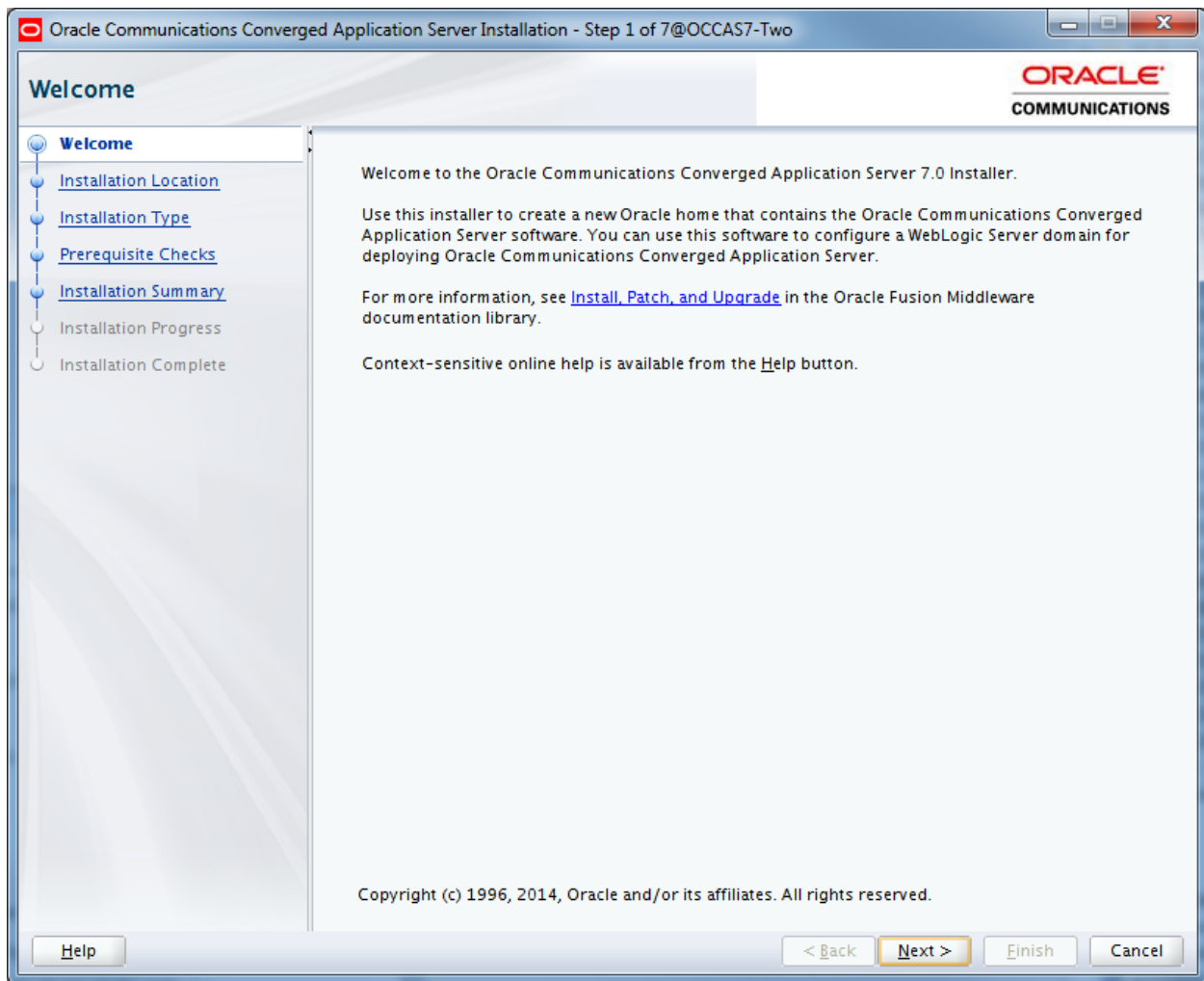
2. Click **OK**.



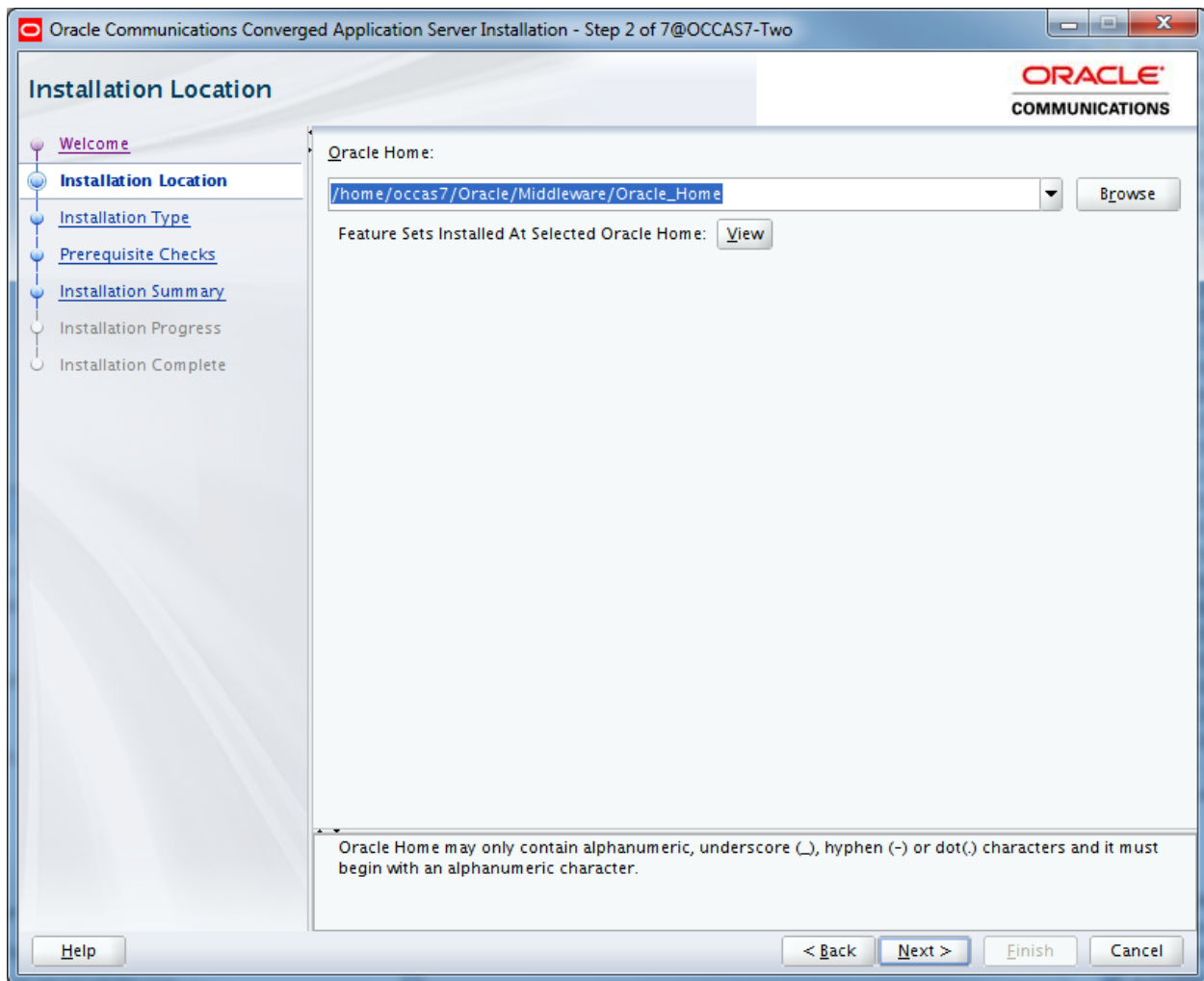
The following screen appears until the wizard sequence finishes generation.



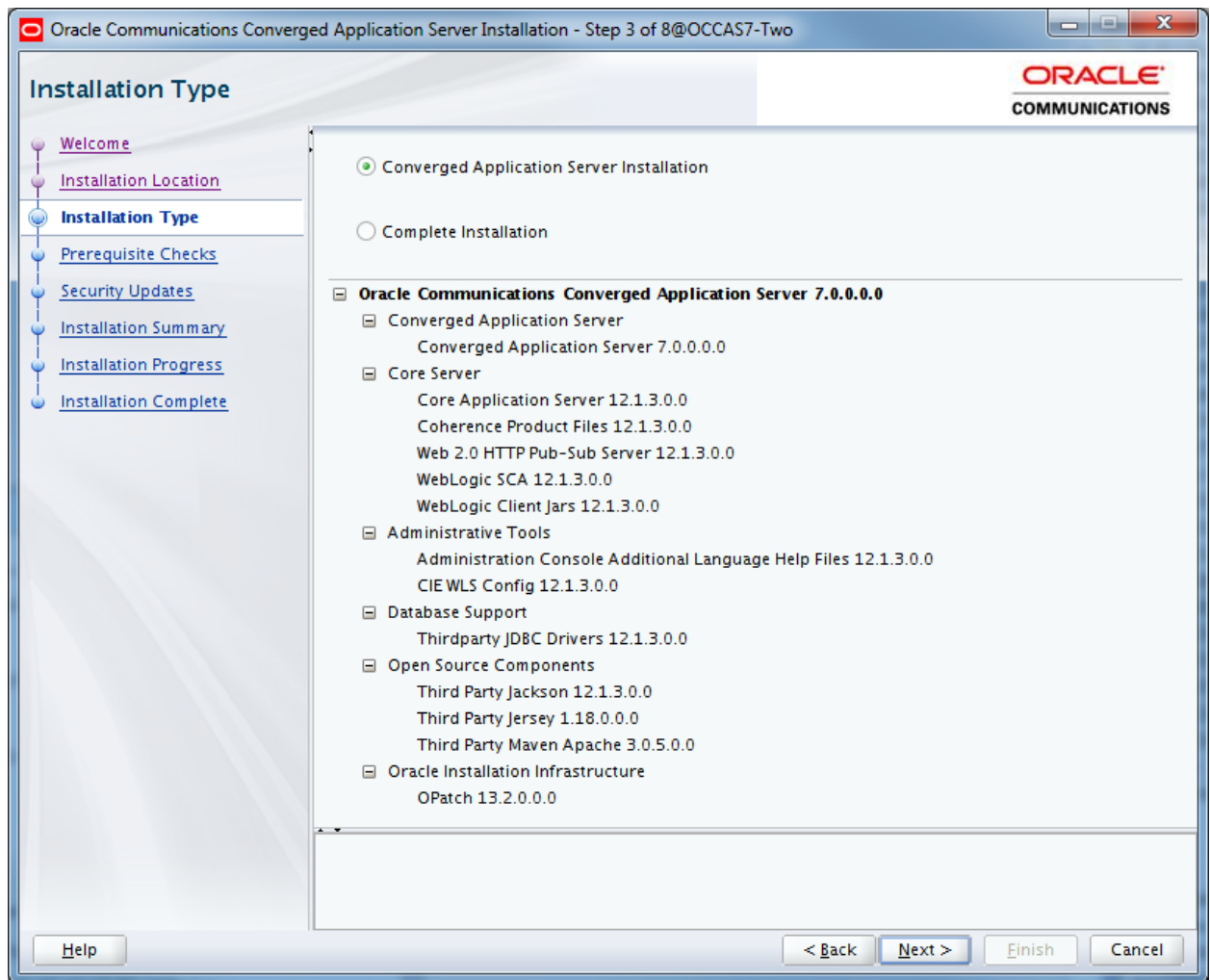
3. Click **Next**.



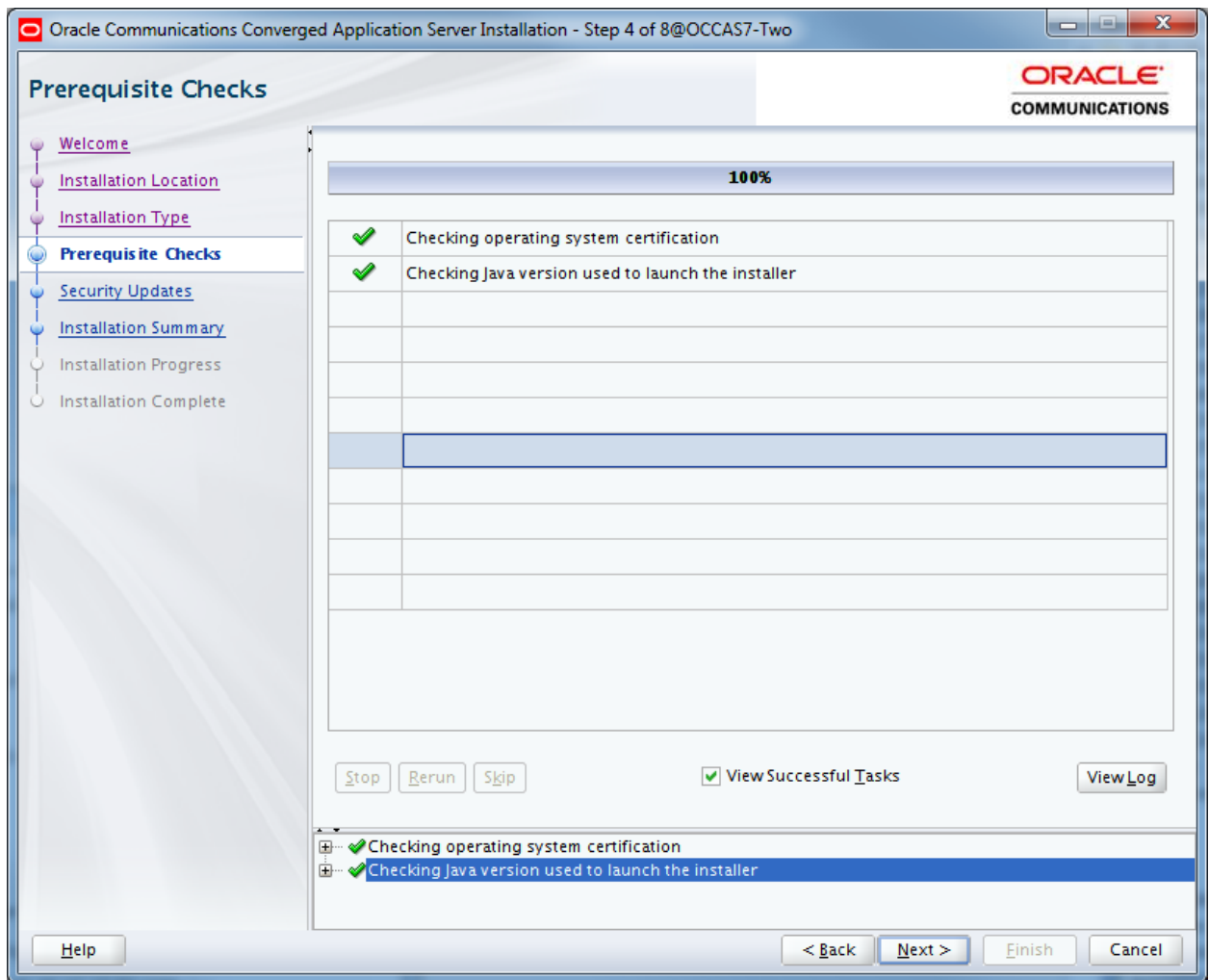
4. Click **Next**.



5. Click **Next**.



6. Click **Next**.



- Unless you have an account with Oracle, deselect **I wish to receive security updates via My Oracle Support**, and then click **Next**.

View details.'. Below this is an 'Email:' label followed by an empty text box. A note says 'Easier for you if you use your My Oracle Support email address/username.' Below that is a checkbox labeled 'I wish to receive security updates via My Oracle Support.' which is currently checked. Below the checkbox is a 'My Oracle Support Password:' label followed by an empty text box. At the bottom of the window are buttons: Help, < Back, Next >, Finish, and Cancel."/>

Oracle Communications Converged Application Server Installation - Step 5 of 8@OCCAS7-Two

Security Updates

Welcome
Installation Location
Installation Type
Prerequisite Checks
Security Updates
Installation Summary
Installation Progress
Installation Complete

Provide your email address to be informed of security issues, install the product and initiate configuration manager. [View details.](#)

Email:

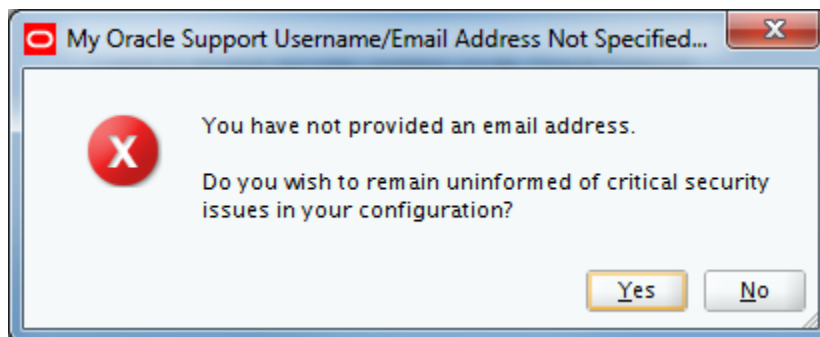
Easier for you if you use your My Oracle Support email address/username.

☒ I wish to receive security updates via My Oracle Support.

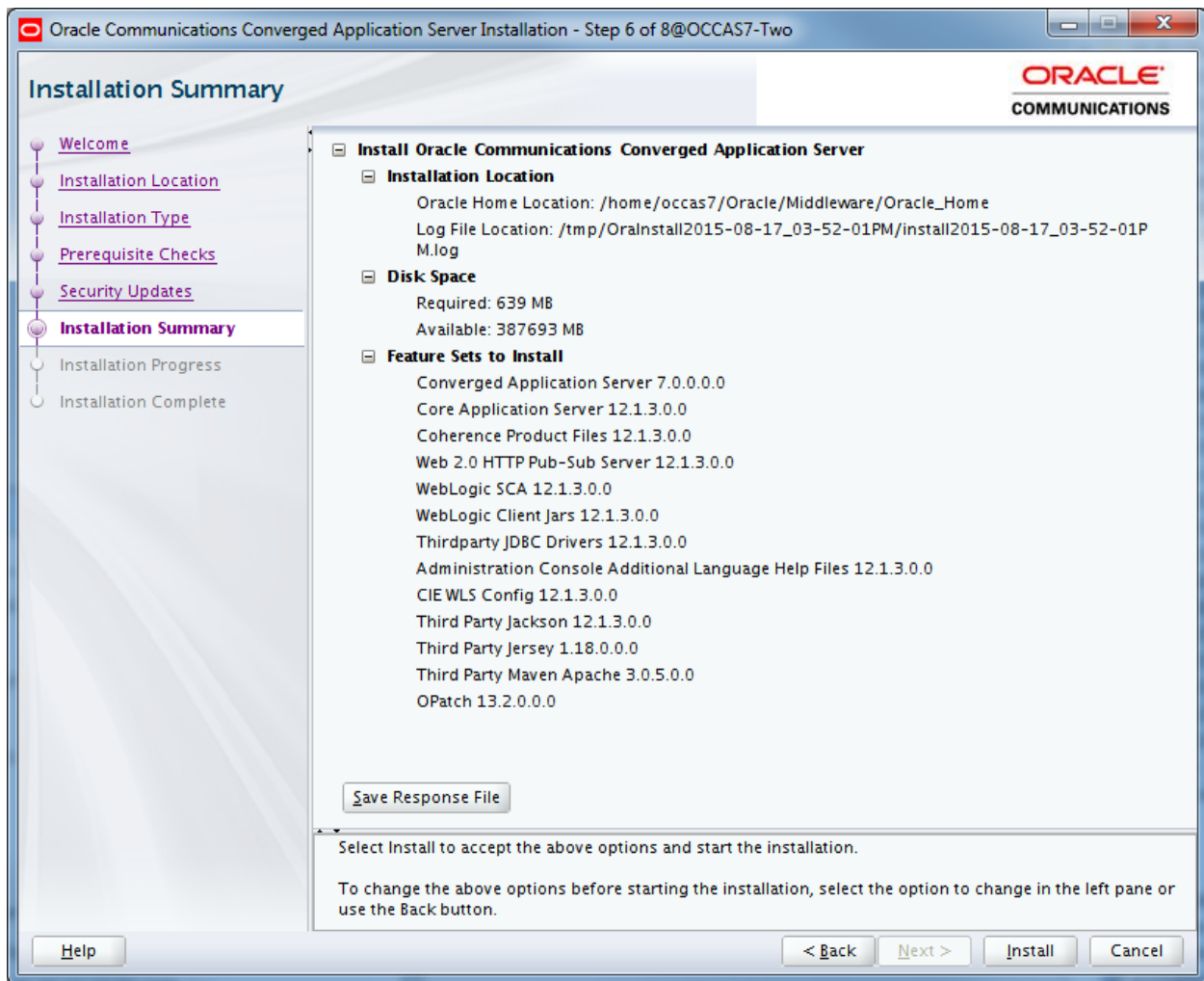
My Oracle Support Password:

Help < Back Next > Finish Cancel

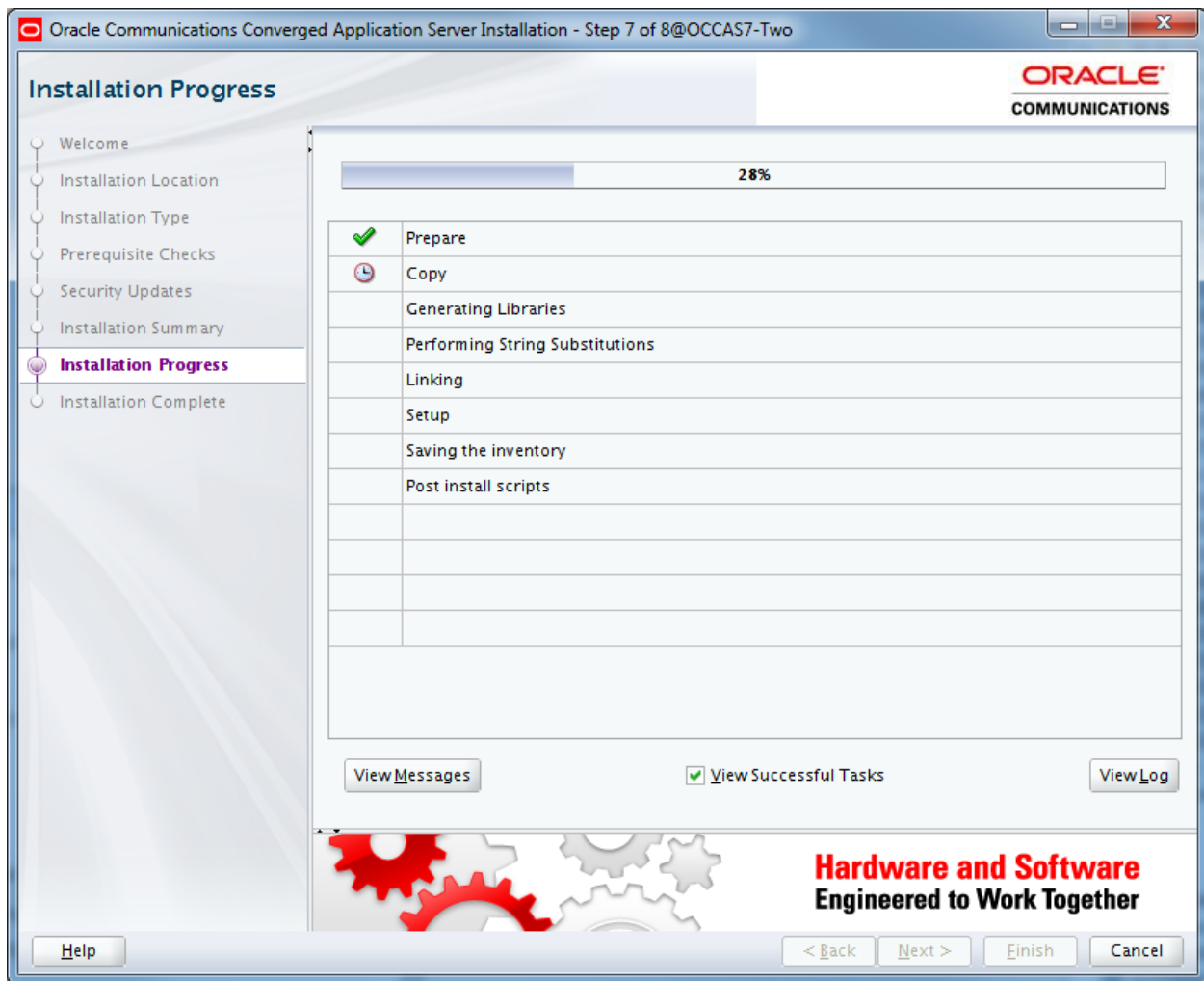
- Click **Yes**.



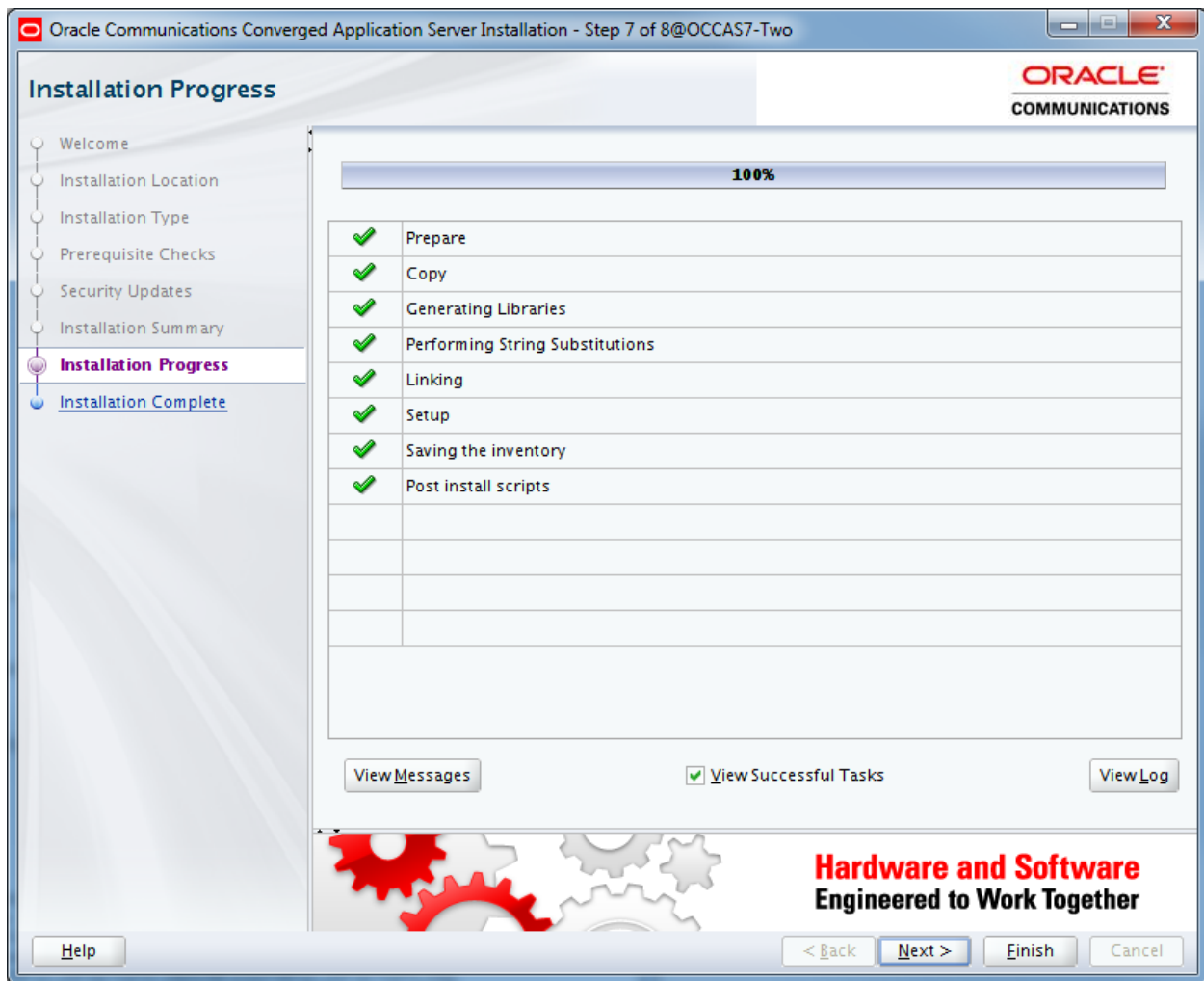
9. Click **Install**.



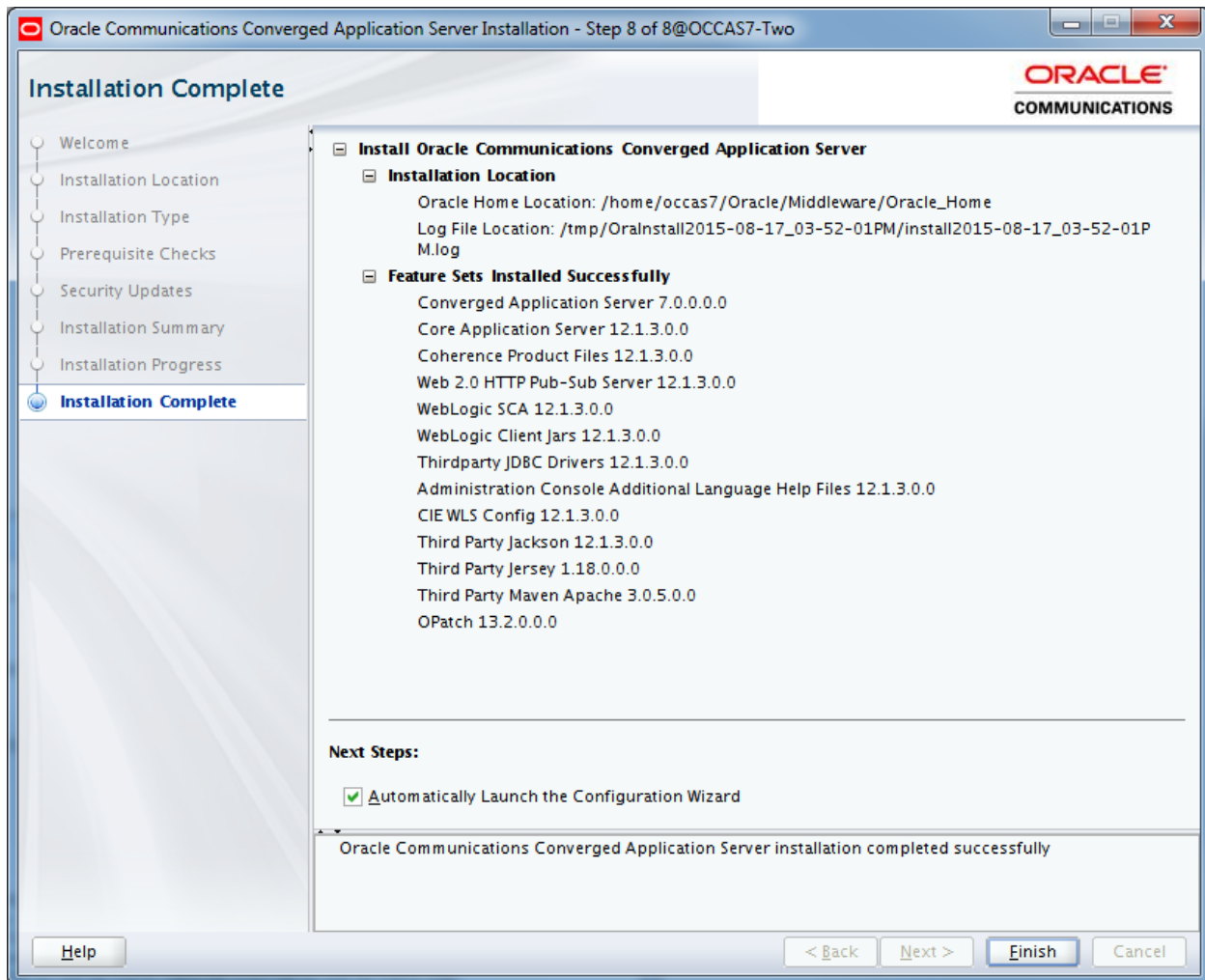
The following screen appears while the installation is in progress.



10. Once completed, click **Next**.



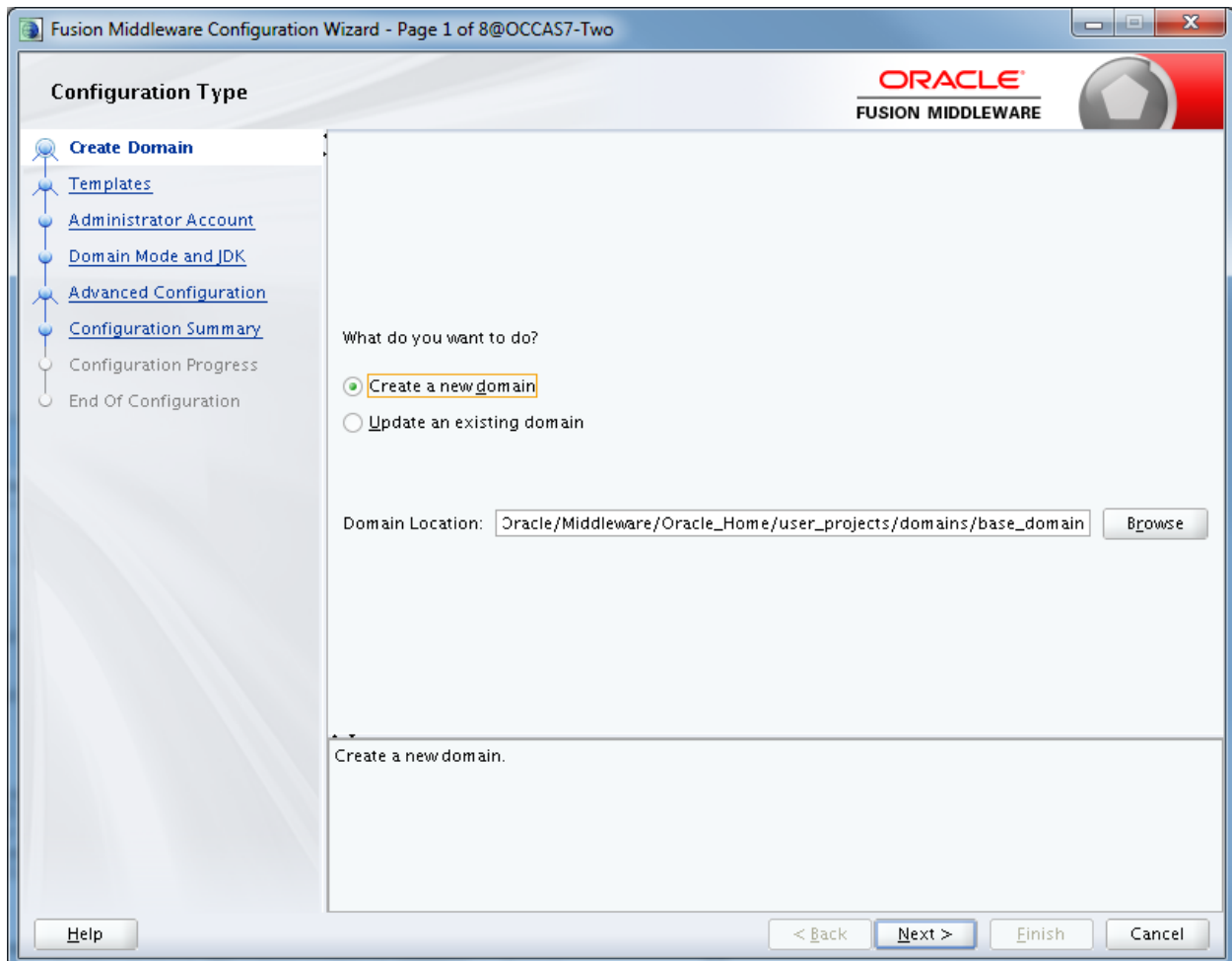
11. After reviewing the installation choices shown on the screen, click **Finish** to automatically launch the configuration wizard. Proceed to OCCAS Configuration.



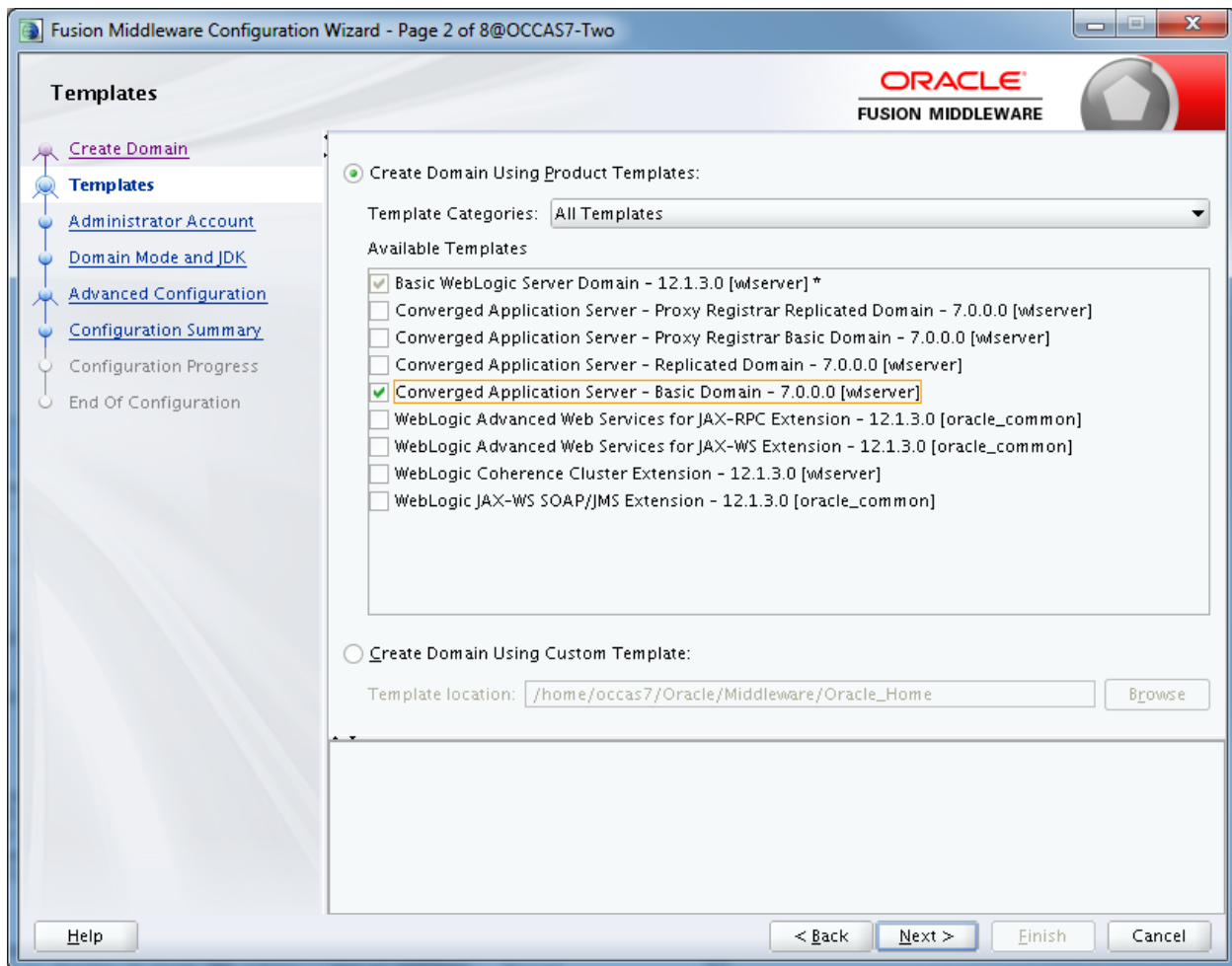
OCCAS Configuration

Proceed as follows to configure OCCASW after installation:

1. Click **Next**.



2. Select **Converged Application Server – Basic Domain**, and then click **Next**.



3. Configure the password for this domain.

Fusion Middleware Configuration Wizard - Page 3 of 8@OCCAS7-Two

Administrator Account

ORACLE
FUSION MIDDLEWARE

Create Domain
Templates
Administrator Account
Domain Mode and JDK
Advanced Configuration
Configuration Summary
Configuration Progress
End Of Configuration

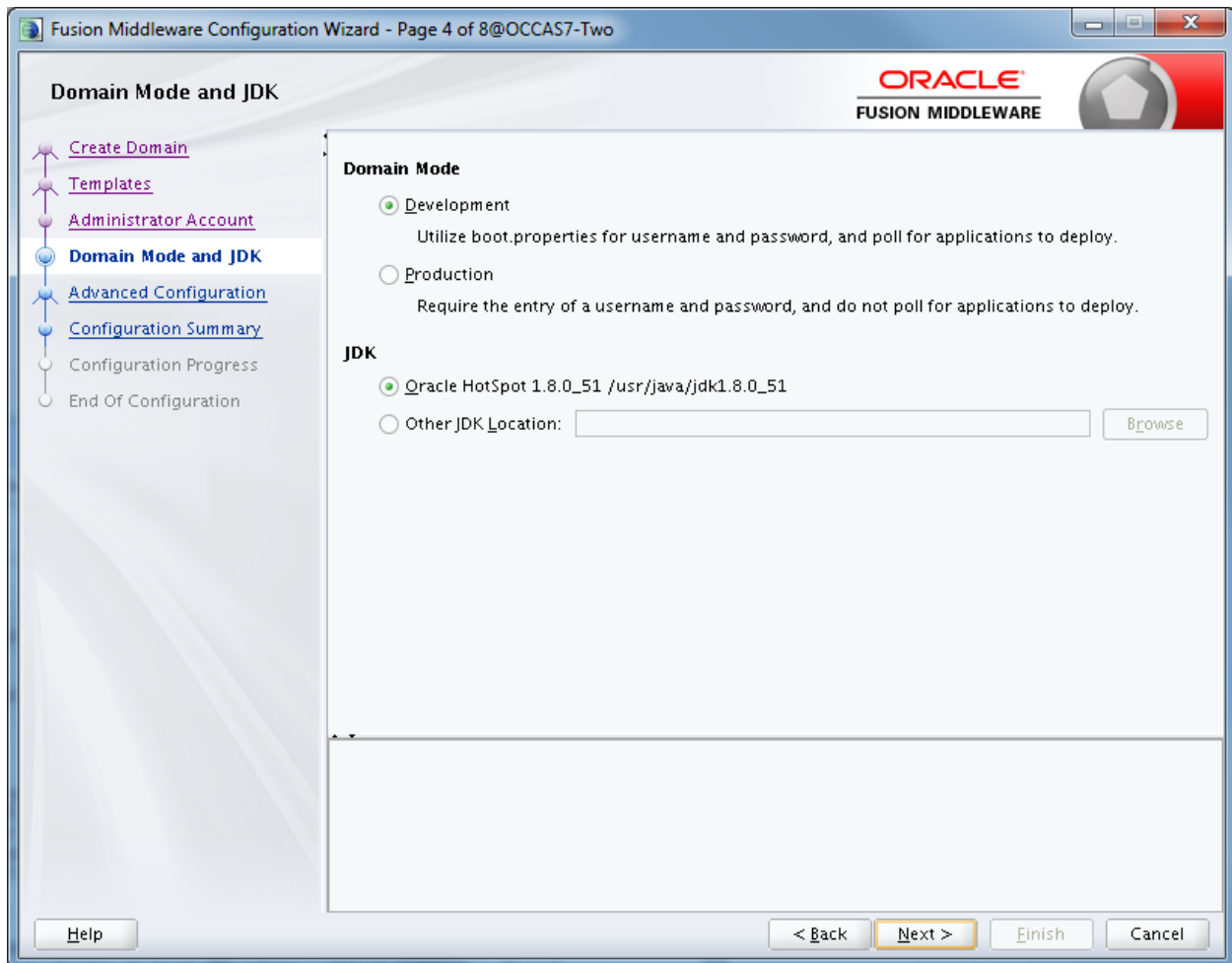
Name

Password

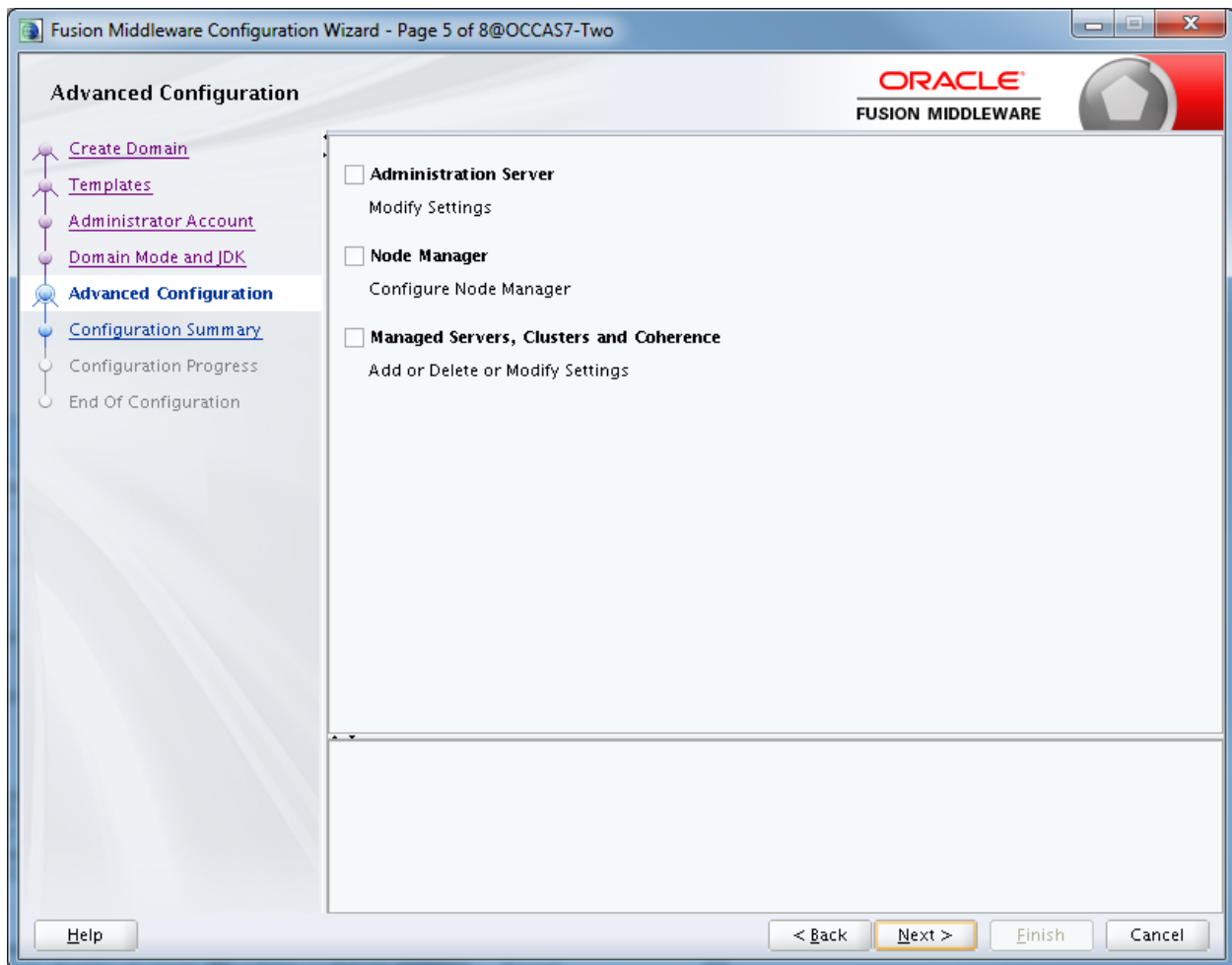
Confirm Password

Help < Back Next > Finish Cancel

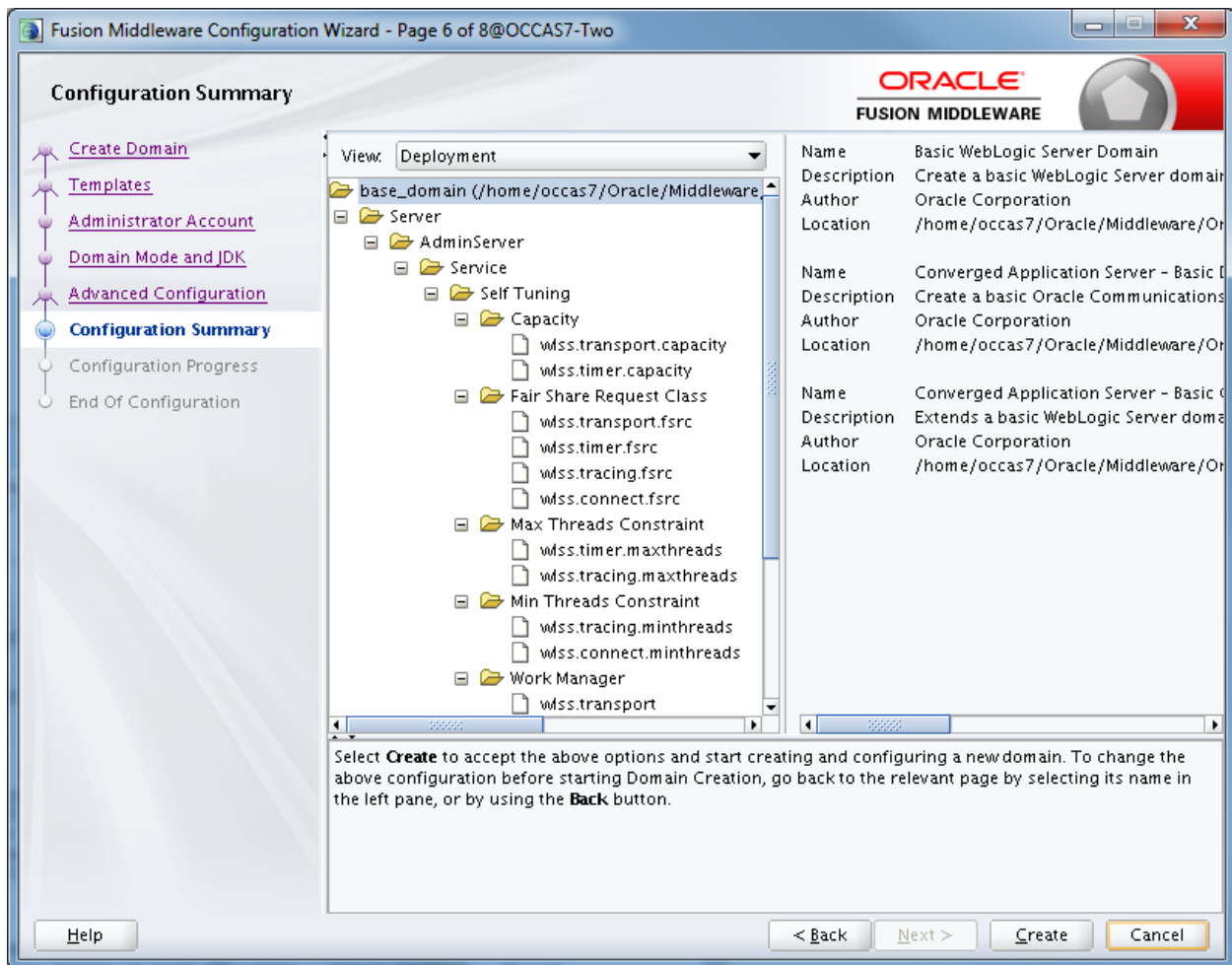
4. Make sure that the JDK points to the correct installation of the JAVA version, and then click **Next**.



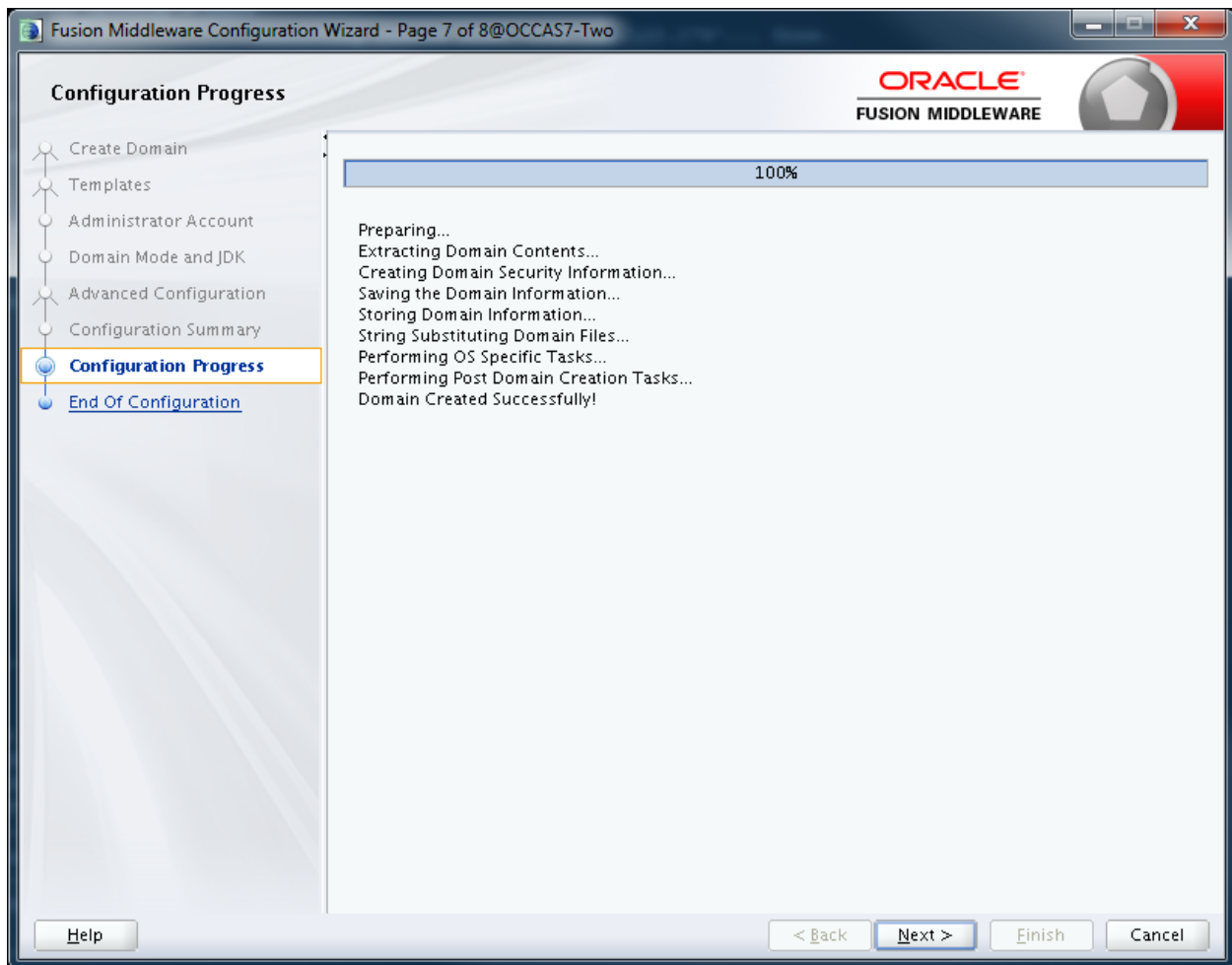
5. Click **Next**.



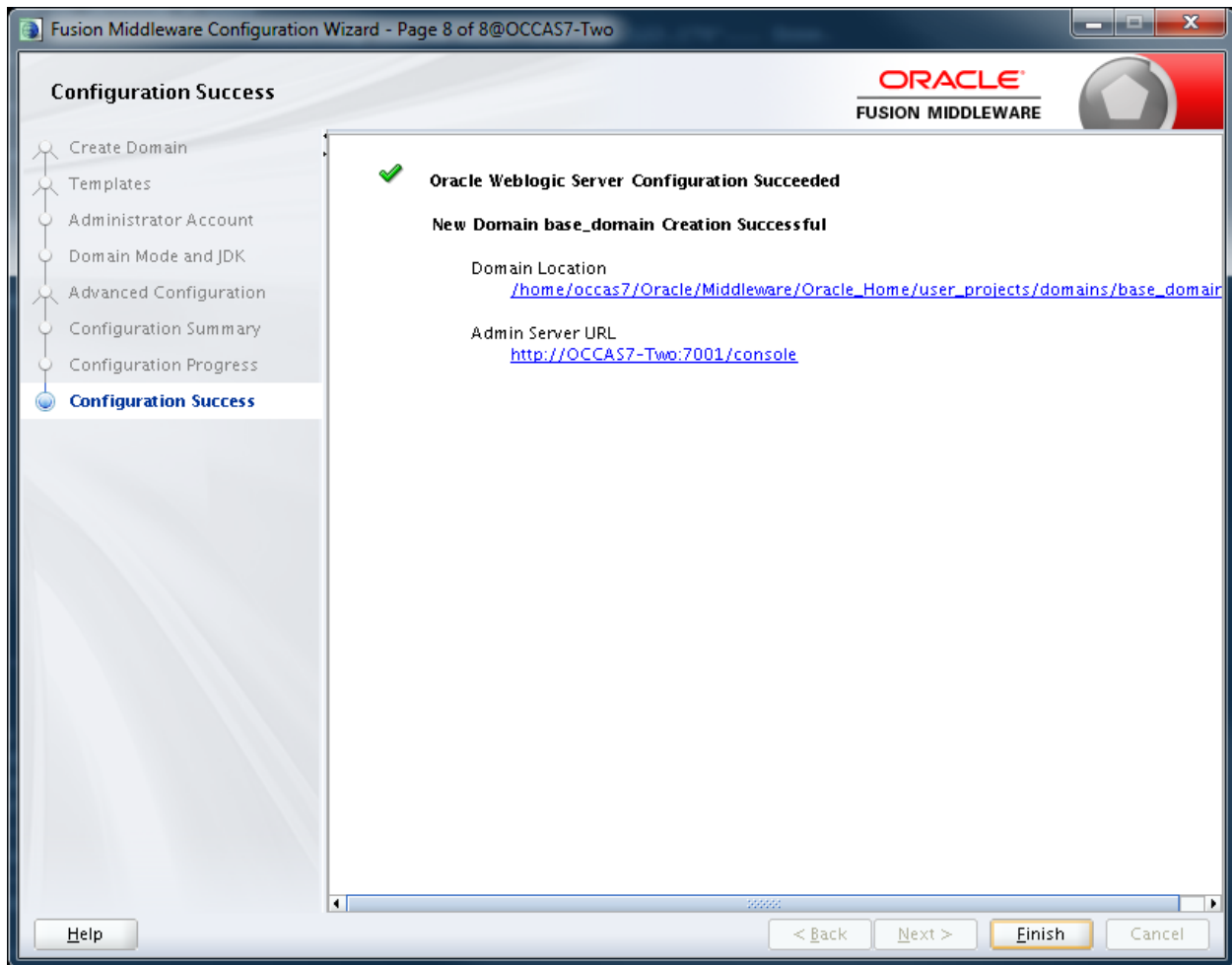
6. Click **Create**.



7. When the domain is created, click **Next**.



8. When configuration is complete, click **Finish**. The OCCAS installation and configuration are now complete.



OCCAS Startup

To start OCCAS 7, go to the `<DOMAIN_HOME>/bin` directory:

```
/home/occas7/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/bin
```

Run the following command:

```
./startWebLogic.sh
```

Since the Development Mode installation was chosen, it is not necessary to enter the username/password during script startup. If the Production Mode installation was chosen, it is necessary to specify the username/password.

The following is an example:

Name: weblogic

User password: Web10gic!! ("0" is a zero)

To verify that OCCAS 7.0 is started, check if **<Server state changed to RUNNING.>** is displayed.

```
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure[1]" is now listening on fe80:0:0:0:7a2b:7002 for protocols iiops, t3s, ldaps, https.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sips[1]" is now listening on fe80:0:0:0:7a2b:7002 for protocols sips.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sip[1]" is now listening on fe80:0:0:0:7a2b:7002 for protocols sip.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sip" is now listening on 146.152.122.192:5060 for protocols sip.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <WebLogicServer> <BEA-000331> <Started WebLogic Admin Server "AdminServer" for development mode>
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport UDP) 60">
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
```

Firewall Configuration

If the firewall is enabled, the 7001 TCP, 5060 UDP, and 5061 UDP IP ports need to be opened in order for OCCAS to work correctly in this scenario. Refer to Oracle OCCAS documentation for further details.

OCCAS Verification

Access the Administration Console to verify the installation at http://<as_ip_address>:7001/console. In the **Domain Structure** section, click **Deployments** to make sure **State** and **Health** are similar to the following screen.

The screenshot displays the Oracle WebLogic Server Administration Console interface. The top navigation bar includes links for Home, Log Out, Preferences, Record, and Help, along with a search bar. The user is logged in as 'weblogic' and connected to the 'base_domain'.

On the left sidebar, the 'Domain Structure' tree is visible, with 'Deployments' selected under the 'base_domain'.

The main content area is titled 'Summary of Deployments' and features a 'Control' tab. It contains a table with columns: Name, State, Health, Type, Targets, and Deployment Order. The table is currently empty, displaying the message 'There are no items to display'.

Below the table, there are controls for 'Install', 'Update', 'Delete', 'Start', and 'Stop'. The status bar at the bottom indicates 'Showing 0 to 0 of 0' items.

On the left sidebar, the 'Change Center' section shows 'View changes and restarts' with a note about configuration editing. The 'How do I...' section lists various tasks such as installing, configuring, updating, and monitoring applications. The 'System Status' section shows the 'Health of Running Servers' with a bar chart indicating 1 OK server.

At the bottom, the footer displays the WebLogic Server version (12.1.3.0.0) and copyright information.

8. Appendix B: Updating the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector comes as a set of JAVA library files (JAR). In the OCCAS 7 Application Server, the required application files are stored as part of the application server configuration and are located in the *lib* directory of the OCCAS 7 *DOMAIN_HOME* directory.

To update the Dialogic JSR 309 Connector library, replace existing (if applicable) JAR files with a new set in the previously referenced *lib* directory.

The Dialogic JSR 309 Connector is a set of the following files:

- *dialogic309-5.x.xxxx-occas7.0.war*
- *dialogic309-5.x.xxxx-occas7.0.jar*
- *dialogicsmiltypes-5.x.xxxx-occas7.0.jar*

To upgrade the existing OCCAS 7.0 Dialogic JSR 309 Connector, there are several steps required:

1. Stop any existing application using the Dialogic JSR 309 Connector.
2. Stop and delete the existing dialogic309-5.x.xxxx-occas7.0 connector application from the OCCAS 7.1 Administration Console.
3. Remove the Dialogic JSR 309 Connector WAR file from following folder:

```
${DOMAIN_HOME}/applications
```

4. Remove previous versions of the two library (JAR) files from the following folder:

```
${DOMAIN_HOME}/lib
```

5. Follow installation instructions to complete the process.