



# **Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 4.1**

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

# Copyright and Legal Notice

---

Copyright © 2014-2015 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](http://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 de la 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, 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 de la 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.

Any use case(s) shown and/or described herein represent one or more examples of the various ways, scenarios or environments in which Dialogic® products can be used. Such use case(s) are non-limiting and do not represent recommendations of Dialogic as to whether or how to use Dialogic products.

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

---

<b>1. JSR 309 Connector Requirements .....</b>	<b>6</b>
<b>2. Contents of the Distribution .....</b>	<b>7</b>
Distributed Files .....	7
<b>3. Installation and Configuration .....</b>	<b>8</b>
Preparing the J2EE Converged Application Server .....	8
Installing the JSR 309 Connector .....	8
Step 1 – Installation and Configuration of JSR 309 Connector .....	8
Configure JSR 309 Connector Required Third-Party Libraries .....	9
Configure JSR 309 Connector Properties File.....	9
Modify AS Startup Script .....	9
Step 2 – Installation and Configuration of JSR 309 Connector Demo .....	10
Configure JSR 309 Connector Demo Properties File.....	10
Modify AS Startup Script .....	10
Deploy JSR 309 Connector Demo Application.....	11
Step 3 – Proper Configuration of PowerMedia XMS .....	20
PowerMedia XMS Web Admin Configuration .....	21
Step 4 – Verification of JSR 309 Connector using Demo Application .....	23
<b>4. Test Servlets .....</b>	<b>24</b>
Test Servlets Overview .....	24
Running the Test Servlets.....	24
DlgcPlayerTest .....	24
DlgcDtmfPromptAndCollectTest .....	24
DlgcRecorderTest .....	25
DlgcDtmfAsyncTest.....	25
Conference Demos .....	26
JMCConferenceServlet.....	26
DlgcReferenceConferenceWithOutBCallServlet .....	27
DialogicBridgeConference .....	28
DlgcEarlyMediaBridgeDemo.....	28
<b>5. Troubleshooting .....</b>	<b>31</b>
Logging .....	31
SIP Errors .....	31
<b>6. Building and Debugging Sample Demos in Eclipse IDE.....</b>	<b>32</b>
Prerequisites .....	32
Creating Build Environment .....	32
Building the Project .....	47
Configuring Eclipse Project and OCCAS Deployed Application for Remote Debugging .....	48
Eclipse Project Remote Debugging Configuration.....	49

<b>7.</b>	<b>Appendix A: JSR 309 Connector Environment Setup .....</b>	<b>52</b>
	Installing and Configuring the OCCAS.....	52
	Pre-Installation Setup .....	52
	JDK Setup .....	53
	OCCAS Installation .....	53
	OCCAS Configuration .....	63
	OCCAS Startup .....	73
	Firewall Configuration .....	75
	OCCAS Verification .....	80
<b>8.</b>	<b>Appendix B: Redundant Media Servers Configuration .....</b>	<b>81</b>
<b>9.</b>	<b>Appendix C: Updating the JSR 309 Connector .....</b>	<b>83</b>

## Revision History

---

Revision	Release Date	Notes
1.3 (Updated)	May 2015	<b>Installation and Configuration:</b> <ul style="list-style-type: none"><li>Added <a href="#">Configuration of WAR File via AS Web Admin Console</a> to the <a href="#">Deploy JSR 309 Connector Demo Application</a> section.</li></ul>
1.3	April 2015	Updates to support JSR 309 Connector Release 4.1. Updates to support log4j2 implementation. <b>JSR 309 Connector Requirements:</b> <ul style="list-style-type: none"><li>Updated the supported version to PowerMedia XMS Release 2.4 Service Update 1.</li></ul> <b>Contents of the Distribution:</b> <ul style="list-style-type: none"><li>Updated the <a href="#">Distributed Files</a> section.</li></ul> <b>Appendix C: Updating the JSR 309 Connector:</b> <ul style="list-style-type: none"><li>Updated with details on the <i>MANIFEST.MF</i> file.</li></ul>
1.2	February 2015	<b>Appendix B: Redundant Media Servers Configuration:</b> <ul style="list-style-type: none"><li>Updated with details on hot active/standby redundancy.</li></ul> <b>Appendix C: Updating the JSR 309 Connector:</b> <ul style="list-style-type: none"><li>Added new section.</li></ul>
1.1	November 2014	Updates with miscellaneous fixes.
1.0	October 2014	Initial version of document.
Last modified: May 2015		

# 1. JSR 309 Connector Requirements

---

The following requirements are needed to be in place before installing the JSR 309 Connector:

- A functional Oracle Communications Converged Application Server (OCCAS) platform for development and testing.

The JSR 309 Connector has been tested with the OCCAS version 5.1.0 Application Server.

- Install required Oracle patch:
  - 18135712 SU Patch [4Q2T]: SIP RP 5.1.0.0.2 - Do Async actions not always running if the action is created within a timer. (Patch)
- A functional PowerMedia XMS Release 2.4 Service Update 1 system.  
**Note:** Refer to [Proper Configuration of PowerMedia XMS](#) for additional information.
- SIP phones or soft clients.

## 2. Contents of the Distribution

---

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

### Distributed Files

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

*Oracle-msc#. #.tar*

This package contains the following structure:

JSR 309 Connector Files	Description
<p><u>DIR:</u> <i>/DlgcJSR309/application/</i></p> <p><u>CONTENTS:</u> <i>deploymentDescriptor/</i> <i>sample-src/</i> <i>build.xml</i> <i>dlgmsc_tests.war</i></p>	<p>Directory that contains a deployable web archive that can be used to test the supported functionality. The WAR file implements several test servlets. Refer to <a href="#">Test Servlets</a> for more information.</p> <p>It also contains the test servlets source files and build environment in order to simply create demo application project.</p>
<p><u>DIR:</u> <i>/DlgcJSR309/lib/</i></p> <p><u>CONTENTS:</u> <i>dlgmsc.jar</i> <i>msmltypes.jar</i> <i>dlgsmiltypes.jar</i> &lt;<i>third-party required files</i>&gt;</p>	<p>Directory that contains the JSR 309 Connector implementation for PowerMedia XMS which consists of a sets of 3 JAR files: <i>dlgmsc.jar</i>, <i>msmltypes.jar</i>, and <i>dlgcsmltypes.jar</i>.</p> <p>Directory also contains additional third-party libraries required by JSR 309 Connector.</p>
<p><u>DIR:</u> <i>/DlgcJSR309/properties/</i></p> <p><u>CONTENTS:</u> <i>dlgc_JSR309.properties</i> <i>dlgc_demos.properties</i> <i>log4j2.xml</i></p>	<p>Directory that contains the properties files used to set up configuration for JSR 309 Connector and provided demos as well as xml configuration file for logging framework.</p>
<p><u>DIR:</u> <i>/DlgcJSR309DemoPrompts/</i></p> <p><u>CONTENTS:</u> <i>prompts.tar</i></p>	<p>JSR 309 Connector prompts used by demo application. Refer to <a href="#">Installing the Demo Prompts</a> for further details.</p>

## 3. Installation and Configuration

---

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

For system requirements and supported platforms, see [JSR 309 Connector Requirements](#).

### Preparing the J2EE Converged Application Server

The JSR 309 Connector has been deployed and tested on OCCAS 5.1.0. If you are **not** familiar with OCCAS or how to set it up, refer to [Appendix: JSR 309 Connector Environment Setup](#) for guidance and detailed instructions on how to get you started.

### Installing the JSR 309 Connector

The JSR 309 Connector is a library used by an application which needs to be configured within Application Server itself.

The JSR 309 Connector demo applications provided with the distribution are used to illustrate some functionality of the JSR 309 Connector. Refer to [Test Servlets](#) section for further details.

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

- [Step 1 – Installation and Configuration of JSR 309 Connector](#)
- [Step 2 – Installation and Configuration of JSR 309 Connector Demo](#)
- [Step 3 – Proper Configuration of PowerMedia XMS](#)
- [Step 4 – Verification of JSR 309 Connector using Demo Application](#)

You need to extract the distribution package as various components (files) will be needed to correctly complete each step. Refer to [Contents of the Distribution](#) which describes the contents in detail.

### Step 1 – Installation and Configuration of JSR 309 Connector

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

```
tar -xvf Oracle-msc#.tar
```

This will create two directories, "*DlgcJSR309*" and "*DlgcJSR309DemoPrompts*", as described in [Contents of the Distribution](#).

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

Follow these steps to properly configure JSR 309 Connector:

- Configure JSR 309 Connector required third-party libraries.
- Configure JSR 309 Connector properties file.
- Modify AS startup script.



## Configure JSR 309 Connector Required Third-Party Libraries

Copy all the extracted content of the distribution package from the "*DlgcJSR309/lib*" directory to the OCCAS "<Domain Location>/lib" directory:

**Note:** "<Domain Location>" refers to the domain path as specified during OCCAS installation.

```
/root/Oracle/Middleware/user_projects/domains/base_domain/lib/
```

## Configure JSR 309 Connector Properties File

JSR 309 Connector is configured by two property files:

- *dlgcsjr309.properties* – used to configure the location (IP addresses and ports) of the OCCAS environment using JSR 309 Connector and of PowerMedia XMS platform.
- *log4j2.xml* – used for connector logging using Simple Logging Facade framework implementation of log4j2.

**Note:** You can configure logging in the *log4j2.xml* file. By default, logging is configured to use INFO level notification and directs logging output to the Console as well as to a separate *dlgmsc.log* file which can be found under "<Domain Location>/bin" directory.

Follow these steps to set up the two properties file:

1. Copy the *dlgcsjr309.properties* and *log4j2.xml* files from "*DlgcJSR309/properties*" directory to the OCCAS "<Domain Location>/config" directory.
2. Edit the *dlgcsjr309.properties* file according to your OCCAS and PowerMedia XMS configuration.
  - The changes will include the OCCAS IP address and port of SipServlet container running the JSR 309 Connector.
  - Changes will also include the PowerMedia XMS IP address and port.

```
...
# Connector's address information (Typically same as the SipServlet container) your OCCAS IP Address
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5060
...
#Media Server
mediaserver.msType=XMS
mediaserver.1.sip.address=xxx.xxx.xxx.xxx
mediaserver.1.sip.port=5060
```

## Modify AS Startup Script

- Edit the <Domain Location>/bin/startWebLogic.sh OCCAS startup script and add the lines in **bold**. Additionally, the OCCAS startup section also needs to be edited in **bold**.

```
.....
JAVA_OPTIONS="${SAVE_JAVA_OPTIONS}"
SAVE_JAVA_OPTIONS=""
CLASSPATH="${SAVE_CLASSPATH}"
#Dialogic additions
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/dlgcsjr309.properties
LOG4J_OPTIONS="-Dlog4j.configurationFile=${DOMAIN_HOME}/config/log4j2.xml"
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/server/lib/jsr309-descriptor-binding.jar"
SERIALIZATION_VALUE=false
SAVE_CLASSPATH=""
trap 'stopAll' 1 2 3 15
.....
```

```

.....
echo "starting weblogic with Java version:"
${JAVA_HOME}/bin/java ${JAVA_VM} -version
if [ "${WLS_REDIRECT_LOG}" = "" ]; then
    echo "Starting WLS with line:"
    echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DEBUG_OPTS} ${LOG4J_OPTIONS} -
Dlog4j.debug -Dwss.local.serialization=${SERIALIZATION_VALUE} -Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS}
${PROXY_SETTINGS} ${SERVER_CLASS}"
    ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DEBUG_OPTS} ${LOG4J_OPTIONS} -Dlog4j.debug -
Dwss.local.serialization=${SERIALIZATION_VALUE} -Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${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} ${DEBUG_OPTS} ${LOG4J_OPTIONS} -Dlog4j.debug -
Dwss.local.serialization=${SERIALIZATION_VALUE} -Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS}
${PROXY_SETTINGS} ${SERVER_CLASS} > "${WLS_REDIRECT_LOG}" 2>&1
fi
stopAll
.....

```

- Now, save the changes and start the WebLogic Server. At this point, the JSR 309 Connector is now configured for a use by an application.

## Step 2 – Installation and Configuration of JSR 309 Connector Demo

At this point, an application can take advantage of JSR 309 Connector and use its resources for media related functionality. The JSR 309 Connector package provides a demo application which uses JSR 309 Connector to illustrate various media functionalities. This step will illustrate how to install and configure JSR 309 Connector demo application. Step 3 will illustrate how to verify that the demo application works with JSR 309 Connector and communicates correctly with PowerMedia XMS.

Follow these steps to set JSR 309 Connector demo application:

1. Configure JSR 309 Connector Demo properties file.
2. Modify AS startup script.
3. Deploy JSR 309 Connector Demo application.

### Configure JSR 309 Connector Demo Properties File

In this step, verify that the demo application works with JSR 309 Connector and communicates correctly with PowerMedia XMS.

- From the distribution package under "*DlgcJSR309/properties*", copy *dlgc\_demos.properties* file to the OCCAS "*<Domain Location>/applications*" directory. The demo properties file has various settings for various sample applications that can be modified. For detailed information on various configurations, refer to the descriptions of each sample application in [Test Servlets](#).

### Modify AS Startup Script

- Edit the *<Domain Location>/bin/startWebLogic.sh* OCCAS startup script and add the lines in **bold**:

```

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

#Dialogic additions
export DIALOGIC_DEMO_PROPERTY_FILE=${DOMAIN_HOME}/applications/dlgc_demos.properties
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/dlgc_JSR309.properties
LOG4J_OPTIONS="-Dlog4j.configurationFile=${DOMAIN_HOME}/config/log4j2.xml"
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/server/lib/jsr309-descriptor-binding.jar"
SERIALIZATION_VALUE=false

SAVE_CLASSPATH=""
trap 'stopAll' 1 2 3 15
.....

```

## Deploy JSR 309 Connector Demo Application

- From the distribution package under the "*DlgcJSR309/application*" directory, copy *dlgmsc\_tests.war* file to the OCCAS "*<Domain Location>/applications*" directory.
- Next, the JSR 309 Connector demo application needs to be deployed in the OCCAS. To do so, proceed to the following instructions.

## Configuration of WAR File via AS Web Admin Console

Make sure that JSR 309 Connector demo application WAR file provided with distribution exists in OCCAS under "*<Domain Location>/application*" directory.

Now, access the **Administration Console** through web browser at:

[http://<as\\_ip\\_address>:7001/console](http://<as_ip_address>:7001/console)

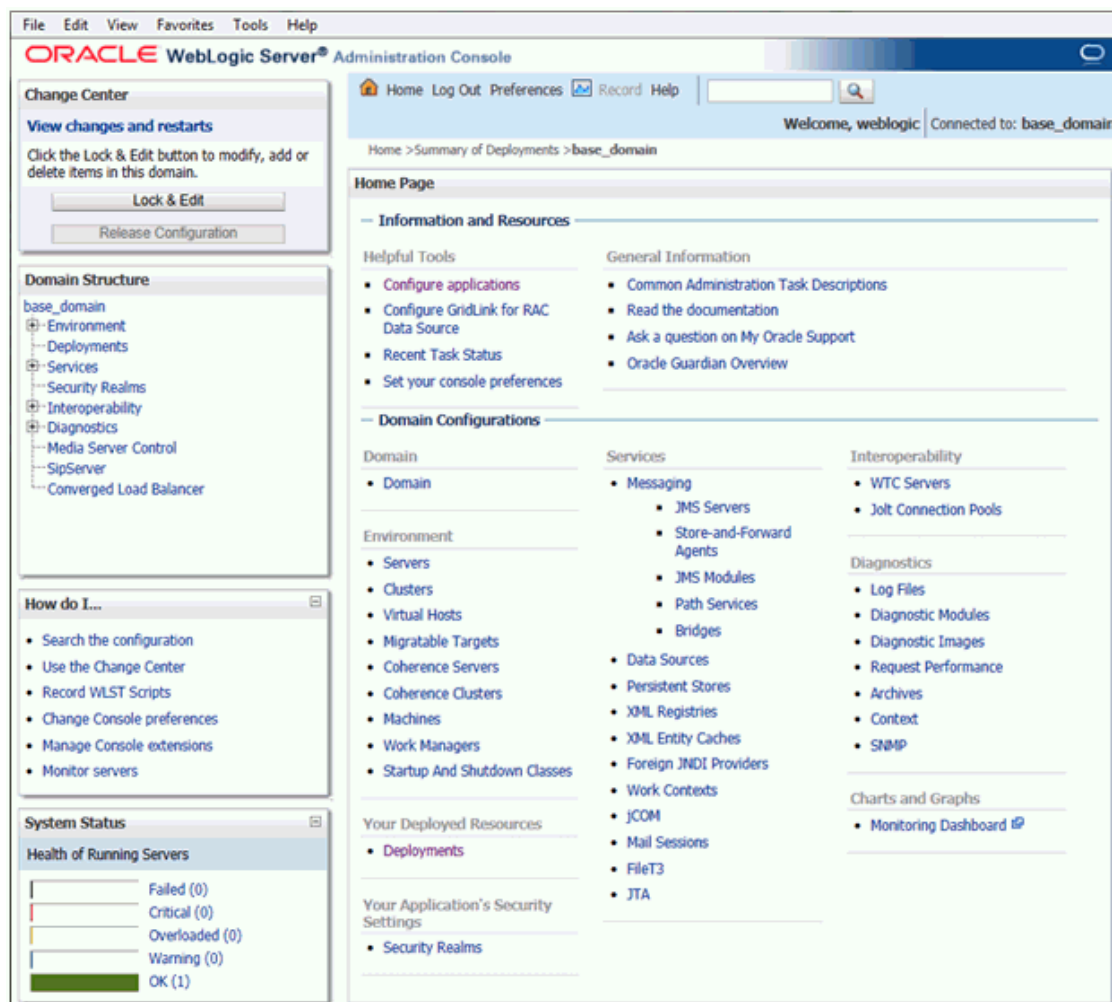
Use the username/password as set during configuration.

The following is used as an example:

**Name:** weblogic

**User password:** Web0gic!! ("0" is a zero)

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



Go to **Deployments** under **Domain Structure**, then click on **Lock & Edit**.

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled "Summary of Deployments" and includes a "Control" tab. Below the tab, there is a text block explaining that the page displays a list of Java EE applications and stand-alone application modules. A "Customize this table" link is present above a table of deployments. The table has columns for Name, State, Health, Type, and Deployment Order. Five entries are listed, all with a State of "Active". The "msrp-connector-ra" entry has a "Health" of "OK".

Name	State	Health	Type	Deployment Order
jax-rs(1.1,1.9)	Active		Library	100
jsr311-api(1.1.1,1.1.1)	Active		Library	100
msrp-connector-ra	Active	OK	Resource Adapter	100
sclb-webglue(5.1.0.0.0,5.1.0.0.0)	Active		Library	100
sft(1.0,1.0)	Active		Library	100

Make sure the existing services are displayed, as shown above, to verify that the OCCAS components have started.

If OCCAS was installed in **Production Mode**, an additional process will need to be followed when making changes to the configuration and setup. This additional process includes **Lock & Edit** to change the configuration and complete the required changes, followed by using **Release Configuration**.

If OCCAS was installed in **Development Mode**, the process is done automatically. Skip the steps that discuss **Lock & Edit**, **Release Configuration**, and activation of an application.

The following examples are based on OCCAS being installed in **Production Mode**.

Click on **Lock & Edit**.

Configuration button to allow others to edit the domain.

Lock & Edit

Release Configuration

**Domain Structure**

- base\_domain
  - Environment
  - Deployments**
  - Services
  - Security Realms
  - Interoperability
  - Diagnostics
  - Media Server Control
  - SipServer
  - Converged Load Balancer

How do I...

**Summary of Deployments**

Control Monitoring

This page displays a list of Java EE applications and this domain. Installed applications and modules can be managed by first selecting the application name and then clicking the appropriate action button.

To install a new application or module for deployment, click the **Install** button.

**Customize this table**

**Deployments**

Install Update Delete Start Stop

<input type="checkbox"/>	Name ^
<input type="checkbox"/>	jax-rs(1.1,1.9)
<input type="checkbox"/>	jsr311-api(1.1.1,1.1.1)
<input type="checkbox"/>	msrp-connector-ra

Click on **Install** under **Deployments**.

Back Next Finish Cancel

**Locate deployment to install and prepare for deployment**

Select the file path that represents the [application](#) root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

**Note:** Only valid file paths are displayed below. If you cannot find your deployment files, [upload your file\(s\)](#) and/or confirm that your [application](#) contains the required deployment descriptors.

Path: /root/Oracle/Middleware/user\_projects/domains/base\_domain/applications

Recently Used Paths: /root/user1

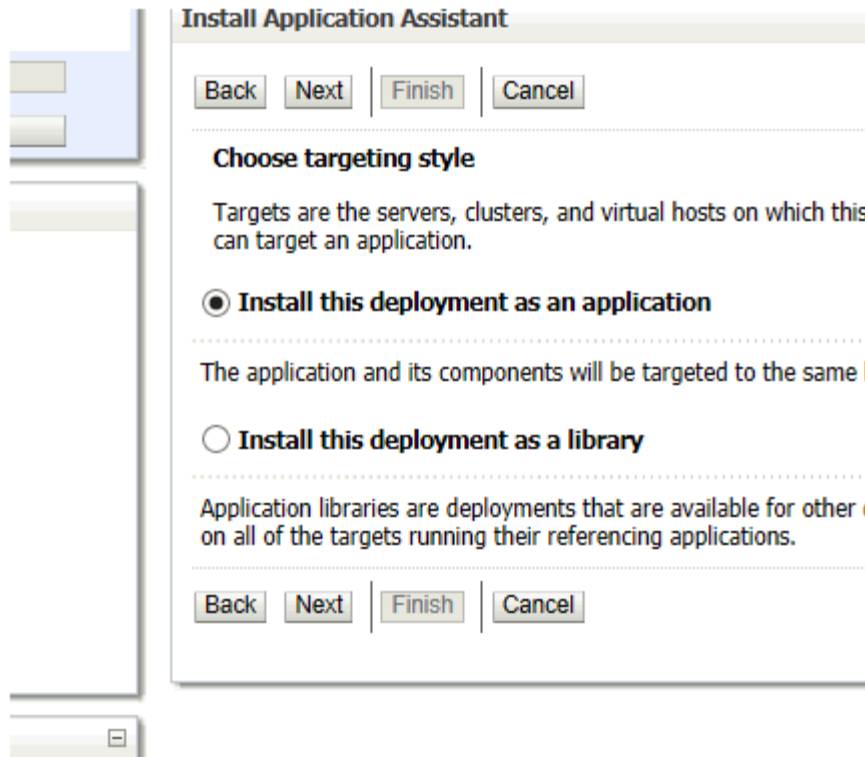
Current Location: 146.152.121.155 / root / Oracle / Middleware / user\_projects / domains / base\_domain / applications

dlgmsc\_tests.war

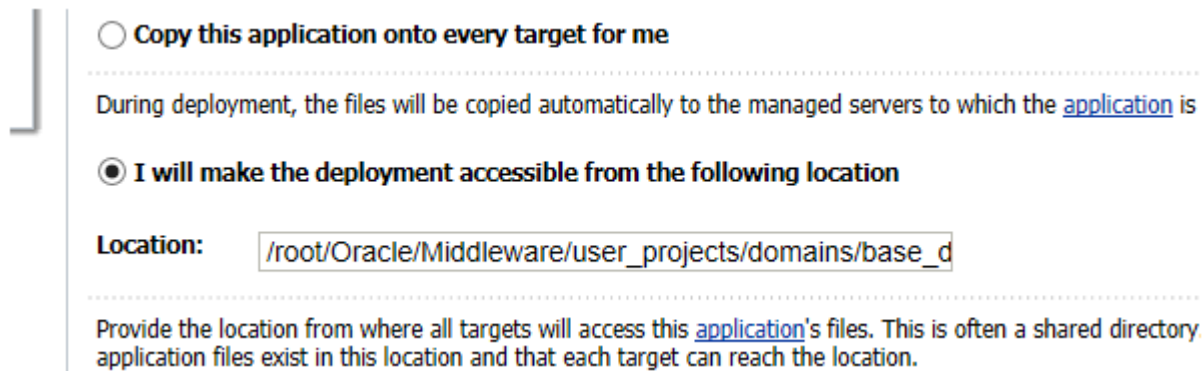
Back Next Finish Cancel

Navigate to "<Domain Location>/applications" under **Current Location**.

Select **dlgmsc\_tests.war**, then click on **Next**.



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



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

## 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 after completing this assistant?

- Yes, take me to the deployment's configuration screen.**
- No, I will review the configuration later.**

### Summary

**Deployment:** /root/Oracle/Middleware/user\_projects/domains/base\_domain/applications/dlgmsc\_tests.war

**Name:** dlgmsc\_tests

**Staging mode:** I will make the deployment accessible at /root/Oracle/Middleware/user\_projects/domains/base\_domain/applications/dlgmsc\_tests.war

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

### Target Summary

Components 	Targets
dlgmsc_tests	AdminServer

[Back](#) [Next](#) [Finish](#) [Cancel](#)

Click on **Finish**.

## Settings for dlgmisc\_tests

**Overview**

Deployment Plan

Configuration

Security

Targets



Control

Testing

Mon

Save

Use this page to view the installed configuration of a Web [Application](#).

<b>Name:</b>	dlgmisc_tests	The name of this <a href="#">applic</a> <a href="#">Info...</a>
<b>Context Root:</b>	/dlgmisc_tests	The specific path at whi found by a servlet. <a href="#">Mc</a>
<b>Path:</b>	/ root/ Oracle/ Middleware/ user_projects/ domains/ base_domain/ applications/ dlgmisc_tests. war	The path to the source the Administration Serv
<b>Deployment Plan:</b>	(no plan specified)	The path to the deploy Administration Server.
<b>Staging Mode:</b>	nostage	The mode that specifies files are copied from a s Administration Server to staging area during app preparation. <a href="#">More Info</a>
<b>Security Model:</b>	DDOnly	The security model spe should be secured. <a href="#">Mc</a>
 <b>Deployment Order:</b>	<input type="text" value="100"/>	An integer value that in deployed, relative to oth server, during startup.
 <b>Deployment Principal Name:</b>	<input type="text"/>	A string value that indic be used when deploying startup and shutdown.

Click on **Save**.



Pending changes exist. They must be activated to take effect.

### Domain Structure

base\_domain

Environment

Deployments

Services

Security Realms

Interoperability

Diagnostics

### Messages

✔ Settings updated successfully.

### Settings for dlgmisc\_tests

Overview

Deployment Plan

Configuration

Sec

Notes

Save

Use this page to view the installed configuration of a

Click on **Activate Changes** once the "Settings updated successfully" message appears.

**Note:** This step is not applicable when OCCAS was installed in **Development Mode**.

ite items in this domain.

### Domain Structure

\_domain

Environment

Deployments

Services

Security Realms

Interoperability

Diagnostics

### Messages

✔ All changes have been activated. No restarts are necessary.

### Settings for dlgmisc\_tests

Overview

Deployment Plan

Configuration

Security

Targets

Co

Notes

Click the **Lock & Edit** button in the Change Center to modify the settings of

Save

Make sure the "All changes have been activated. No restarts are necessary" message appears.

### Domain Structure

base\_domain

Environment

Deployments

Services

Security Realms

Interoperability

Diagnostics

Media Server Control

SipServer

Converged Load Balancer

Over

Note

Click

Save

Use

Nam

Cont

Go back to **Deployments** under **Domain Structure**.

The screenshot shows the Oracle WebLogic Server Administration Console interface. The main content area is titled "Summary of Deployments" and includes a "Control" tab. Below the tab, there is a descriptive paragraph and a "Customize this table" link. A table titled "Deployments" lists the following items:

<input type="checkbox"/>	Name	State	Health	Type	Deployment Order
<input type="checkbox"/>	dlgmsc_tests	Prepared	OK	Web Application	100
<input type="checkbox"/>	jax-rs(1.1,1.9)	Active		Library	100
<input type="checkbox"/>	jsr311-api(1.1.1,1.1.1)	Active		Library	100
<input type="checkbox"/>	msrp-connector-ra	Active	OK	Resource Adapter	100
<input type="checkbox"/>	sclb-webglue(5.1.0.0.0,5.1.0.0.0)	Active		Library	100
<input type="checkbox"/>	sft(1.0,1.0)	Active		Library	100

At the bottom of the console, the "System Status" section shows the "Health of Running Servers" with a bar chart indicating 1 OK server and 0 servers in Failed, Critical, Overloaded, or Warning states.

The **Deployments** along with its **State** and **Health** are displayed. Verify that the installed *dlgmsc\_tests.war* application is in **Prepared** state.

**Deployments**

Install Update Delete Start Stop Showing 1 to

<input type="checkbox"/>	Name	Health	Type
<input checked="" type="checkbox"/>	dlgmisc_tests	Prepared	Web Appli
<input type="checkbox"/>	jax-rs(1.1,1.9)	Active	Libra
<input type="checkbox"/>	jsr311-api(1.1.1,1.1.1)	Active	Libra
<input type="checkbox"/>	msrp-connector-ra	Active	Reso Adap
<input type="checkbox"/>	sclb-webglue(5.1.0.0.0,5.1.0.0.0)	Active	Libra
<input type="checkbox"/>	sft(1.0,1.0)	Active	Libra

Install Update Delete Start Stop Showing 1 to

Once in **Prepared** state, select *dlgmisc\_tests.war* application and click on **Start**, then click on **Servicing all requests**.

**Note:** This step is not applicable when OCCAS was installed in **Development Mode**.

the Lock & Edit button to modify, add or e items in this domain.

Lock & Edit

Release Configuration

ain Structure

- domain
- wironment
- eployments
- ervices
- ecurity Realms
- interopability
- iagnostics

Home > Summary of Deployments > base\_domain > Summary of Deployments > dlgmisc\_tests > Summary of De

**Start Application Assistant**

Yes No

**Start Deployments**

You have selected the following deployments to be started. Click 'Yes' to continue, or 'No' to cancel.

- dlgmisc\_tests

Yes No

Click on **Yes**.

Messages

✔ Start requests have been sent to the selected Deployments.

Summary of Deployments

Control Monitoring







This page displays a list of Java EE applications and stand-alone application modules that have this domain. Installed applications and modules can be started, stopped, updated (redeployed), the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install

▶ Customize this table

Deployments

Install Update Delete Start Stop Showing 1 to 6 of 6

<input type="checkbox"/>	Name	State	Health	Type
<input type="checkbox"/>	 dlgmisc_tests	Active	✔ OK	Web Application
<input type="checkbox"/>	 jax-rs(1.1,1.9)	Active		Library
<input type="checkbox"/>	 jsr311-api(1.1.1,1.1.1)	Active		Library
<input type="checkbox"/>	 msrp-connector-ra	Active	✔ OK	Resource Adapter
<input type="checkbox"/>	 sclb-webglue(5.1.0.0.0,5.1.0.0.0)	Active		Library
<input type="checkbox"/>	 sft(1.0,1.0)	Active		Library

Install Update Delete Start Stop Showing 1 to 6 of 6

Make sure the "Start requests have been sent to the selected Deployments" message appears. Verify that deployed *dlgmisc\_tests.war* application is in **Active** state.

### Step 3 – Proper Configuration of PowerMedia XMS

In order to verify the correct JSR 309 Connector installation with provided JSR 309 Connector demos, you will need to correctly configure PowerMedia XMS Media Server. This includes:

- PowerMedia XMS Web Admin Configuration.
  - Allowing Absolute Paths
  - Installing JSR 309 Connector Demo Prompts
- Demo required prompts installed on PowerMedia XMS itself (optional).

**Note:** Only needed if JSR 309 Connector demo application is going to be used as it depends on these prompts to be installed on PowerMedia XMS.

## PowerMedia XMS Web Admin Configuration

### Allowing Absolute Paths

JSR 309 Connector uses Native MSML interface to PowerMedia XMS Media Server. You need to verify that PowerMedia XMS is indeed configured for "Native" mode.

**Note:** In PowerMedia XMS Release 2.1 and later, "Native" mode is the mode configured by default when PowerMedia XMS gets installed. Also, it is strongly recommended that the latest version of PowerMedia XMS be used.

General		Services	Mode	Time	Backup/Restore	Upgrade	NFS Mount
<b>XMS</b>							
release						2.1.5695	
mode						native	
state						RUNNING	
<b>System</b>							

Now, under the **Media** menu, click on **Media Configuration** tab. The **Allow Absolute Paths** field must be set to YES.

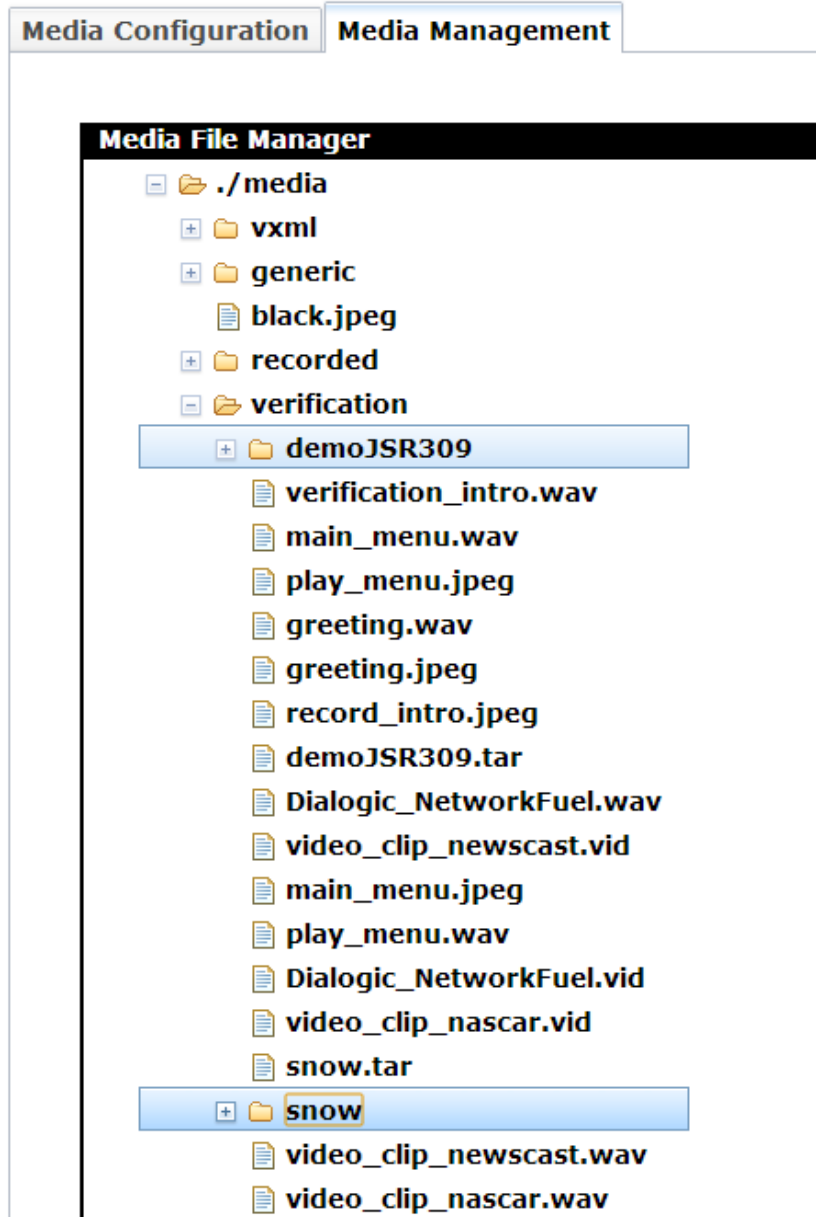
System	<b>Media Configuration</b>	Media Management
Network	Media File Path:	/var/lib/xms/media
License	Locale:	en-US
MSML	Allow Absolute Paths:	YES
MRCP Client	<input type="button" value="Apply"/>	
HTTP Client		
VXML		
RESTful API		
Protocol		
Codecs		
Routing		
Tones		
<b>Media</b>		

Once appropriate changes are made, click the **Apply** button which will commit the changes. Once changes are applied, you will be asked to restart PowerMedia XMS. This step is not yet required since we are going to be changing more configuration parameters below.

## Installing JSR 309 Connector Demo Prompts

Custom prompts need to be installed on PowerMedia XMS as the various Dialogic demos will require them to work.

Once installed, they should appear in the **Media** menu under the **Media Management** tab as shown below in the highlighted fields:



You can locate and install the demo prompts by performing the following:

1. Copy the *prompts.tar* file inside the "*<Release Package>/DlgcJSR309DemoPrompts*" directory to the PowerMedia XMS system under the "*/var/lib/xms/media/en\_US/verification*" directory.
2. Untar file using the command:

```
tar -xvf <file_name>.tar
```

## Step 4 – Verification of JSR 309 Connector using Demo Application

With default *dlgc\_demos.properties* file, you can use a simple ***DlgcPlayerDemo*** to verify the proper connector installation and operation. This demo simply answers an incoming call and uses JSR 309 Connector to request media resources from PowerMedia XMS in order to play an audio file.

Here are the steps on running ***DlgcPlayerDemo*** for verification.

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

```
DlgcPlayerDemo@<as_ip_address>
```

3. With successful configuration, user should hear a verification prompt being played out.

## 4. Test Servlets

---

This section describes the test servlets (basic sample applications) and requirements for running test servlets in the JSR 309 Connector.

### Test Servlets Overview

Test Servlets, or sample applications, are included as part of distribution. They illustrate the use of the JSR 309 Connector. These test servlets are included in the *dlgmsc\_tests.war*. For installation instructions and any additional requirements for running test servlets, refer to [Installation and Configuration of JSR 309 Connector Demo](#) and to [Proper Configuration of PowerMedia XMS](#).

### Running the Test Servlets

When using any standard SIP phone a special SIP URI will be used to initiate each test servlet.

#### DlgcPlayerTest

This test servlet plays a PowerMedia XMS pre-set prompt.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcPlayerDemo**. Make sure that the Web Application Server is running the *dlgmsc\_tests.war* application.

Using the demo property file, set the following:

```
player.test.prompt=
```

For example:

```
player.test.prompt=file:///var/lib/xms/media/en_US/verification/greeting.wav
```

The player will play this prompt. Make sure that the prompt file exists in the Media Server.

To test the application, dial the following:

```
DlgcPlayerDemo@<as_ip_address>
```

#### DlgcDtmfPromptAndCollectTest

This test servlet plays a prompt and collects DTMF digits.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcPromptCollectDemo**. Make sure that the Web Application Server is running the *dlgmsc\_tests.war* application.

The **DlgcPromptCollectDemo** can be controlled using the demo property file as follows:

- The `detectOnlyTest` reads the number of signals property value and sends the pattern x (times number of signals). Note that no prompt is played. The following example generates a pattern to match of any five (5) DTMF entries:

```
signalDetector.test=detectOnlyTest  
signalDetector.number_of_signals=5
```

- The `detectPromptCollectTest` plays a prompt and looks for a given pattern. It does not make use of the number of signals property.

```
signalDetector.test=detectPromptCollectTest  
signalDetector.match_pattern=min=1;max=5;rtk=#
```



- The `detectCollectWithPatternTest` does not prompt the user and only uses the `match_pattern`.

```
signalDetector.test=detectCollectWithPatternTest
signalDetector.match_pattern=min=1;max=5;rtk=#
```

**Note 1:** You can configure the signal detector with the following properties (for example, the timeout values are based in milliseconds units):

```
signalDetector.initial_digit_timeout=5000
signalDetector.inter_digit_timeout=5000
signalDetector.max_duration=10000
```

**Note 2:** For the test that plays a prompt, you can control a loop (or how many times the test repeats the prompt and collect) by controlling the following property:

```
signalDetector.loopCounter=2
```

To test the application, dial the following:

```
DlgcPromptCollectDemo@<as_ip_address>
```

## DlgcRecorderTest

This test servlet records a greeting.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcRecorderDemo**. Make sure that the Web Application Server is running the `dlgmsc_tests.war` application.

In the SIP phone, select your newly created test contact. You are prompted to record your greeting at the tone. After the tone, say your greeting, and enter **#000** to play your greeting.

After the greeting is played back, the application completes by hanging up the phone. If you do not enter **#000**, the greeting continues to record until the timeout is reached.

The recording demo can be controlled in the demo property file by configuring the following record properties:

```
record.test.file=file:///tmp/recorder jsr309 test demo.ulaw
record.test.minDuration=6000
record.test.maxDuration=60000
record.test.initialTimeout=7000
record.test.finalTimeout=4000
record.test.silenceTerminationFlag=true
```

To test the application, dial the following:

```
DlgcRecorderDemo@<as_ip_address>
```

## DlgcDtmfAsyncTest

This test servlet illustrates the asynchronous DTMF capabilities.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcAsyncDtmfDemo**. Make sure that the Web Application Server is running the `dlgmsc_tests.war` application.

In the SIP phone, select your newly created test contact. Notice that there are no prompts. You will be connected. The application waits for you to press DTMF digits. For each DTMF pressed, the application will receive the DTMF and print the collected DTMF to the screen.

Selecting the number **0** hangs up the connection.

To test the application, dial the following:

```
DlgcAsyncDtmfDemo@<as_ip_address>
```

## Conference Demos

The following table depicts the conference demos that are delivered with JSR 309 Connector:

Demo Name	Functionality	Requires
JMCConferenceServlet	Demonstrates how to create and manage multiple conferences.	Media files need to be installed in the PowerMedia XMS for menu to work.
DlgcAvLayoutConferenceDemo	Implements an advanced conference.	Media files need to be installed in the PowerMedia XMS for menu to work. The demo property file must be configured.
DialogicBridgeConference	Shows how to create a two leg conference without using a mixer.	Media files need to be installed in the PowerMedia XMS for menu to work.

### JMCConferenceServlet

This test servlet illustrates how to create and manage multiple conferences using a mixer control leg. A mixer control leg is an extra SIP connection used to control the conference mixer.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcMultiConferenceDemo**. Make sure that the Web Application Server is running the *dlgmsc\_tests.war* application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This conference performs the following:

1. Establishes a network connection and joins it with a media group.
2. Plays a prompt for a new number (conference pin) and collects signals. Any pin number can be provided. Initially no conferences exist. Conferences are created as users call in and provide pin numbers. Callers will only hear other callers who provide the same pin number.
3. Creates a conference if a new pin is used, or adds a leg to an existing conference.

To test the application, dial the following:

```
DlgcMultiConferenceDemo@<as_ip_address>
```

## DlgcReferenceConferenceWithOutBCallServlet

This test servlet illustrates how to implement an advanced conference that does not require a mixer control leg. The legs are connected directly into a conference without requiring any initial IVR functionality.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcAvLayoutConferenceDemo**. Make sure that the Web Application Server is running the `dlgmsc_tests.war` application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This simple conference performs the following:

1. Can join multiple legs into a conference.
2. Once in conference, the user can enter **\*00** to hear the conference menu and apply some of the menu options.

The demo can be controlled by configuring the following properties in the demo property file:

- Change the initial direction of legs by entering the following properties in the application demo property file:

```
demos.join.direction.leg1=<duplex,recv,send>
demos.join.direction.leg1=<duplex,recv,send>
demos.join.direction.leg1=<duplex,recv,send>
```

- To make an outbound call, make sure you have another accessible SIP phone that can receive calls and configure the following attributes:

```
application.sipTOA_Address.sip.address=146.152.245.3 # IP address of the SIP Phone
application.sipTOA_Port.sip.port=5060
application.sipTOA.sip.username=kapanga #(any name will do)
application.early_media_bridge.sip.address=146.152.122.127 #OCCAS Addr
application.early_media_bridge.sip.port=5060 #OCCAS SIP PORT
```

- To run a video conference, make sure you set the following configuration:

```
media.mixer.mode=AUDIO_VIDEO # possible values AUDIO,AUDIO_VIDEO
media.mixer.conf.video.size=VGA # possible values VGA, 720p
media.mixer.conf.recordfile=file:///tmp/confRecording # recording the conference file
full path. This also works for audio only conference.
```

**Note:** To play the conference recording after the recording is completed, change the following attribute to point to the recording path:

```
player.test.prompt=file:///tmp/confRecording then run DlgcPlayerDemo
```

To test the application, dial the following:

```
DlgcAvLayoutConferenceDemo@<as_ip_address>
```

## DialogicBridgeConference

This test servlet illustrates how to implement a simple conference that does not require a mixer, and that has two legs directly joined into it.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcBridgeDemo**. Make sure that the Web Application Server is running the *dlgmsc\_tests.war* application.

In the SIP phone, select your newly created test conference contact. Notice that you will need at least two SIP phones. The first connection entering the conference will not hear anything until the other legs join in.

This simple conference performs the following:

- Joins two calling legs into a simple conference.
- In order for the leg to enter the bridge, each leg must enter **\*03** after making the call.

To test the application, dial the following:

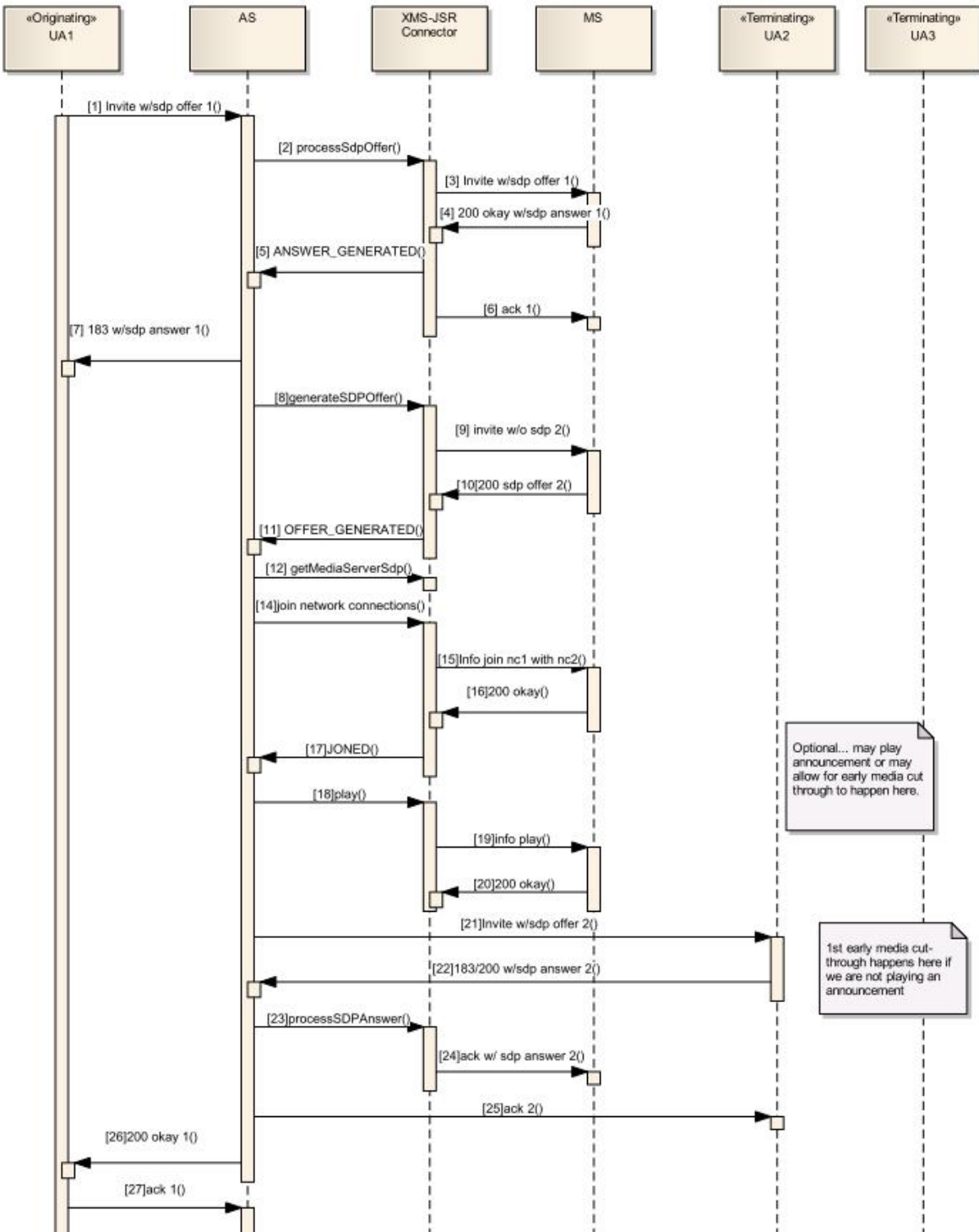
```
DlgcBridgeDemo@<as_ip_address>
```

## DlgcEarlyMediaBridgeDemo

This test servlet is similar to the DialogicBridgeConference defined above, except that it simulates an early media scenario.

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e., URI) to **DlgcEarlyMediaBridgeDemo**. Make sure that the Web Application Server is running the *dlgmsc\_tests.war* application.

The following sequence diagram illustrates **DlgcEarlyMediaBridgeDemo**:



Menu supported by **DlgcEarlyMediaBridgeDemo**:

**\*00** – Plays announcement of menu options

**\*77** – Plays announcement of how the demo works

**\*88** – Plays announcement informing the user if the application is in a bridge or mixer conference

**\*99** – Transfers the two call leg from a bridge conference to a full conference using a mixer

**Note:** Once in a mixer conference, the test application does not allow you to go back to a bridge conference. The following property configuration must be set for this demo to work:

```
application.sipTOA Address.sip.address=146.152.245.3 # IP address of the SIP phone
application.sipTOA Port.sip.port=5060
application.sipTOA.sip.username=kapanga # (any name will do)
application.early_media_bridge.sip.address=146.152.122.127 #OCCAS Addr
application.early_media_bridge.sip.port=5060 #OCCAS SIP PORT
```

To test the application, dial the following:

```
DlgcEarlyMediaBridgeDemo@<as_ip_address>
```

## 5. Troubleshooting

---

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

### Logging

The JSR 309 Connector and sample applications generate log output to the *dlgmsc.log*. The default logging level is set to INFO.

You may also need to enable logging for SIP messages in the container so that the incoming requests that trigger the servlets are captured. You can enable SIP message logging or any other platform related logging through the Application Server administration console.

### SIP Errors

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

- Verify the available PowerMedia XMS licenses.
- Check the */etc/hosts* file configuration.
- Make sure application properties file (i.e., *dlgc\_demos.properties*) is referencing the appropriate PowerMedia XMS and Application Server IP address and ports.

## 6. Building and Debugging Sample Demos in Eclipse IDE

---

The 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 provided demo application using Eclipse IDE.

### Prerequisites

User will need to have installed the following components:

- JDK 1.6.0\_45

**Note:** Latest version of 1.6 JDK is used because this is a version supported by OCCAS 5.1.0.

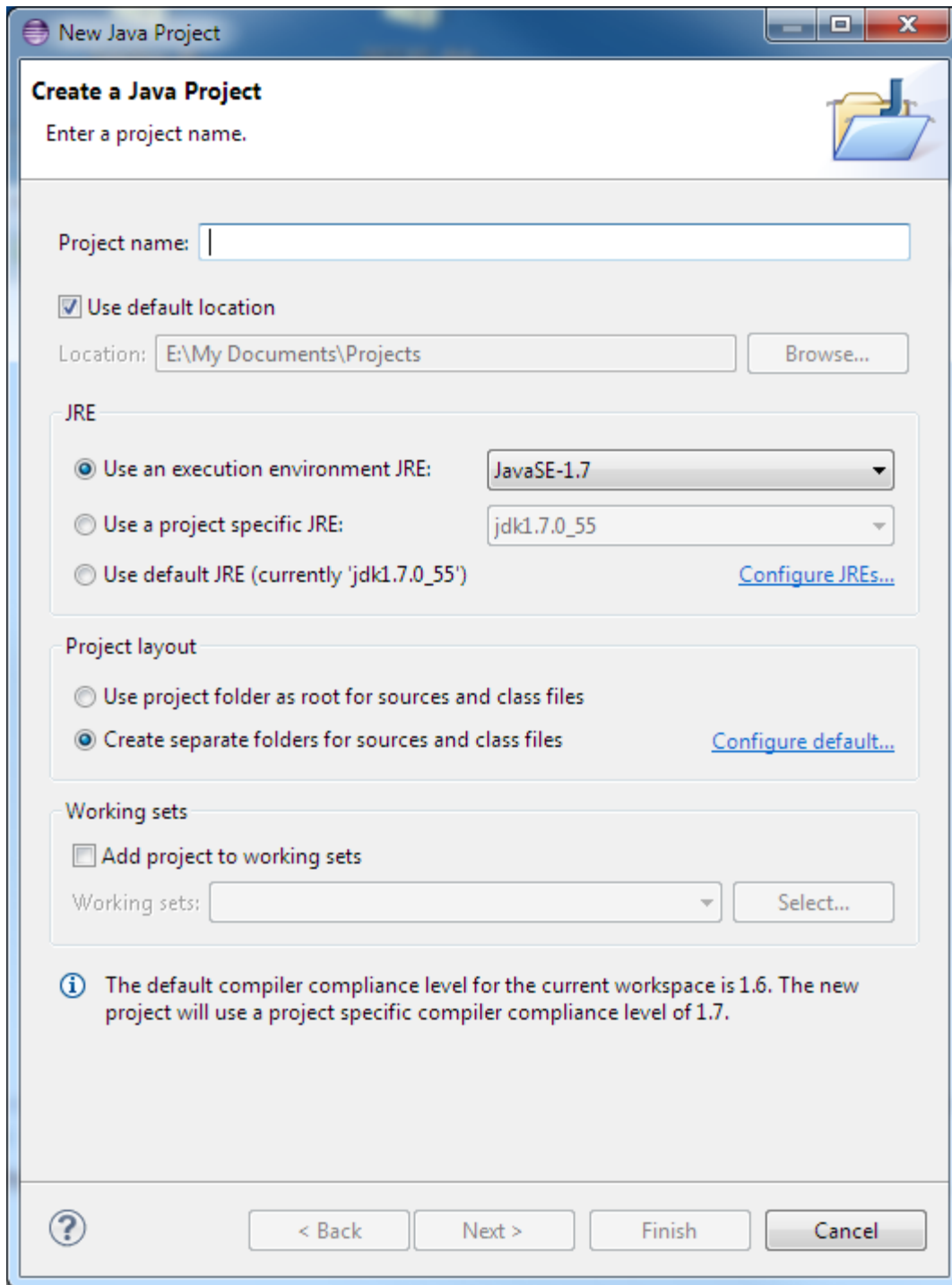
- Eclipse KDE (Eclipse Standard SDK – Kepler Service Release 2 used here).
- In order to build provided demo applications, you will need to obtain two OCCAS 5.1.0 libraries which are NOT provided with JSR 309 Connector distribution:
  - *javax.servlet\_1.0.0.0\_2-5.jar*
  - *wlss.jar*

### Creating Build Environment

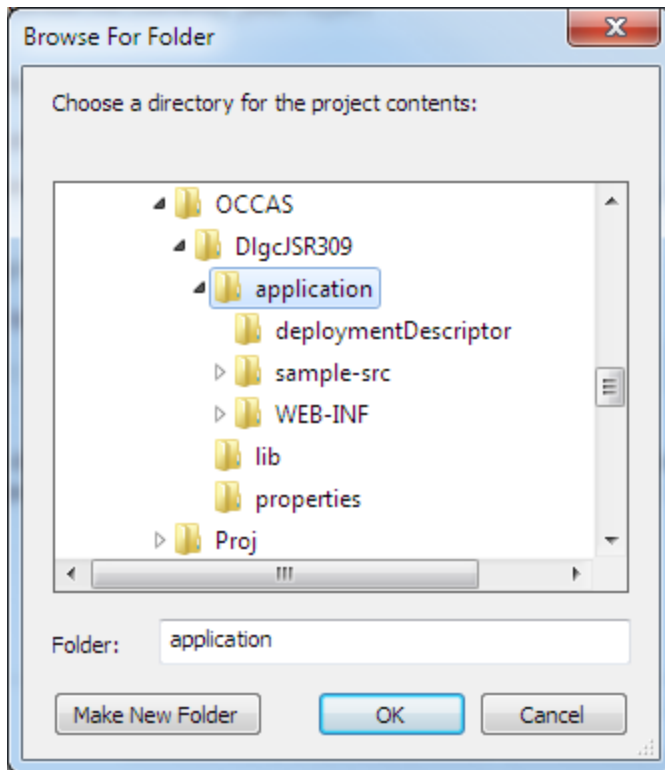
Follow these steps to create a Dialogic demo build environment:

1. From the distribution package, copy the "*DlgcJSR309*" directory and its content to a known location on your system.
2. Copy the required Application Server Platform specific libraries into the "*DlgcJSR309/lib*" directory.
3. Open **Eclipse IDE** and go to **File > New > Java Project**. The following window will appear:

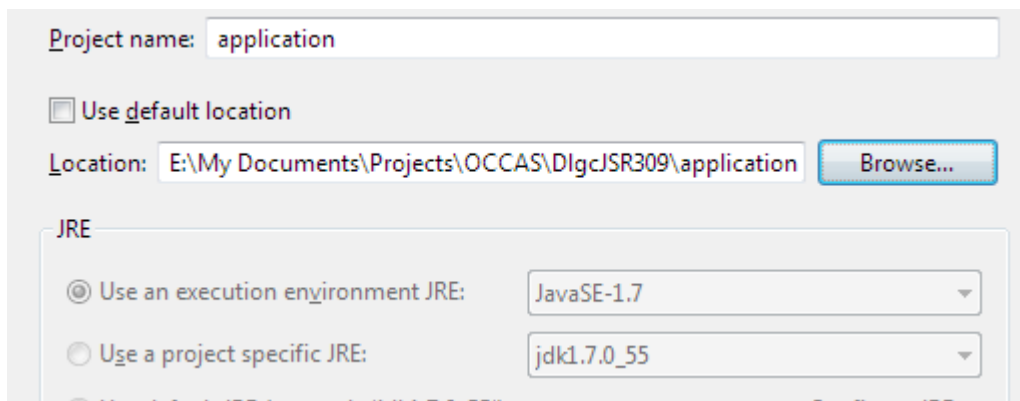




4. Uncheck **Use default location** and then click the **Browse** button.



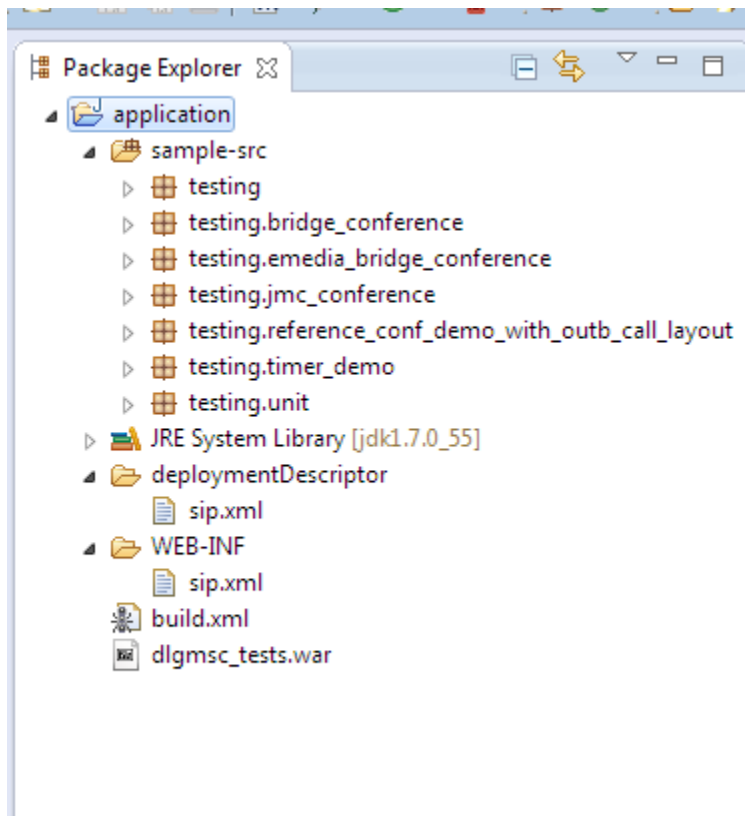
5. Browse to the location of the copied "DlgcJSR309" directory and select the **application** directory. Then, click **OK**.



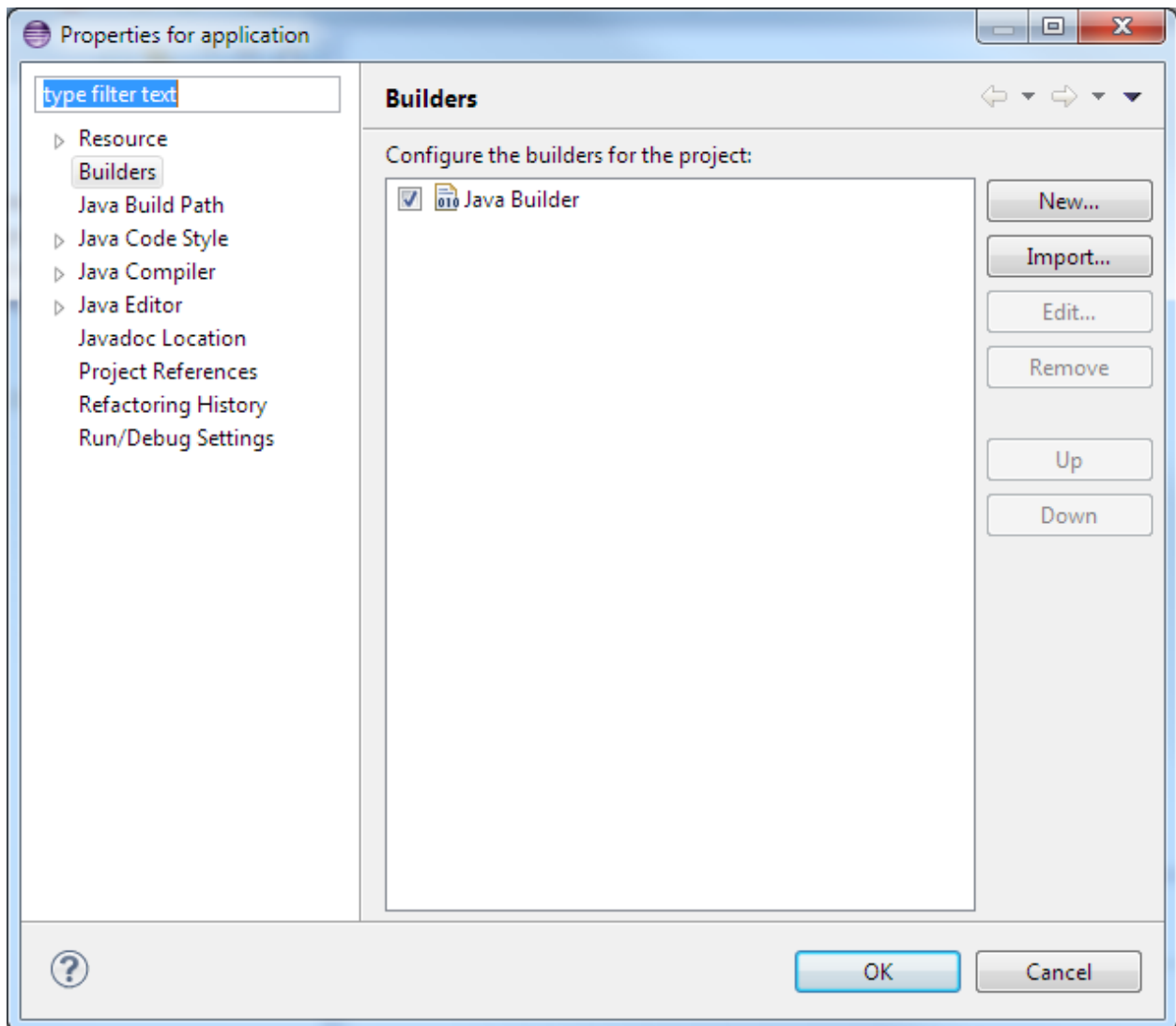
**Note:** Enter any **Project name** you wish to use in the **Project name** field.

6. Now, click **Finish**.

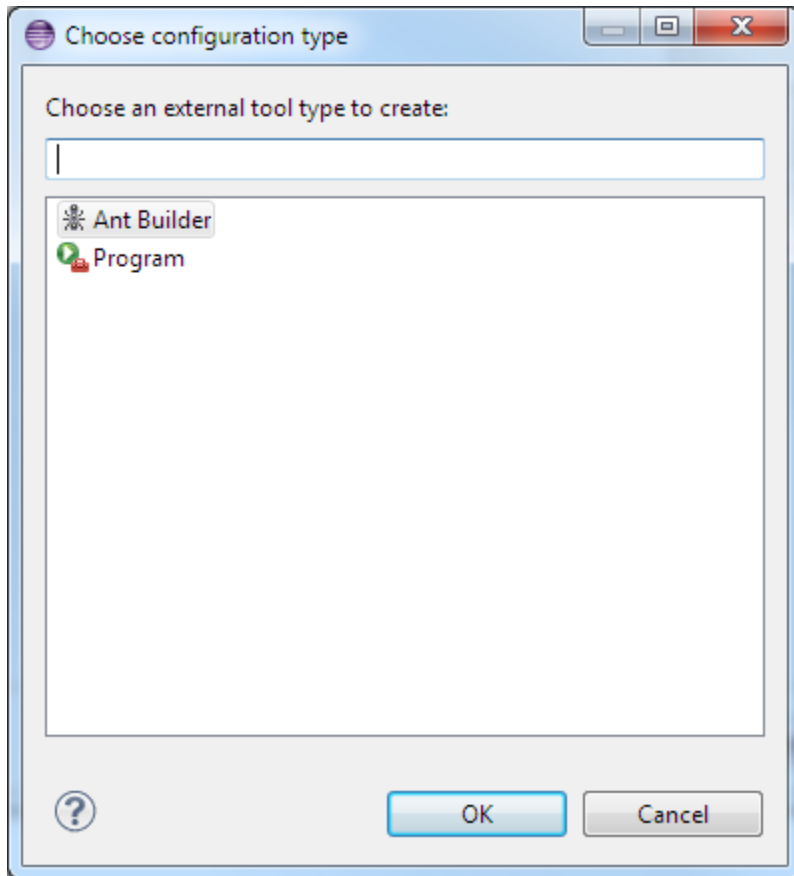
Expanding the project in Eclipse should give you the following content:



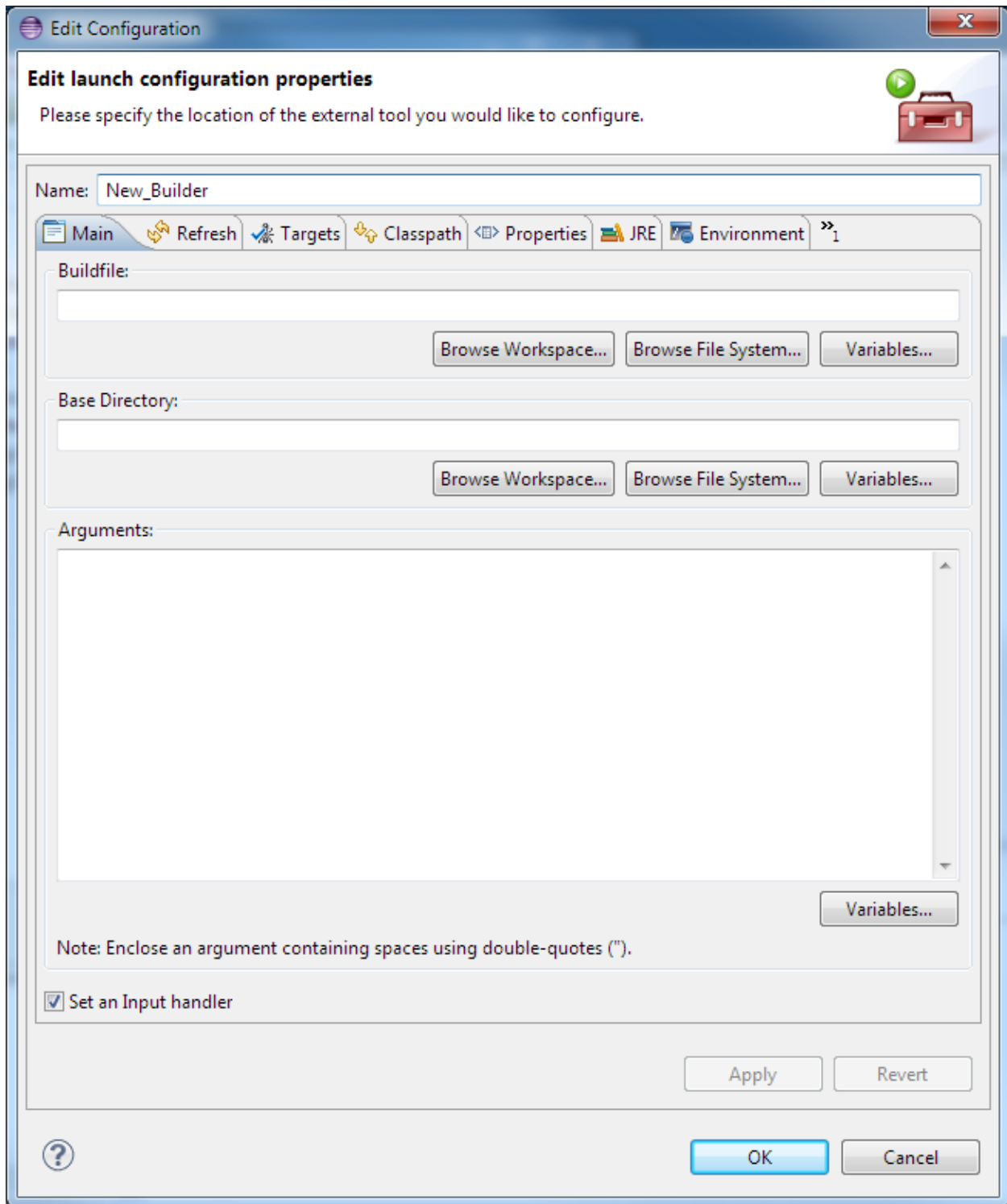
Next, right click on the name of your project in the **Package Explorer** view and select **Properties**.



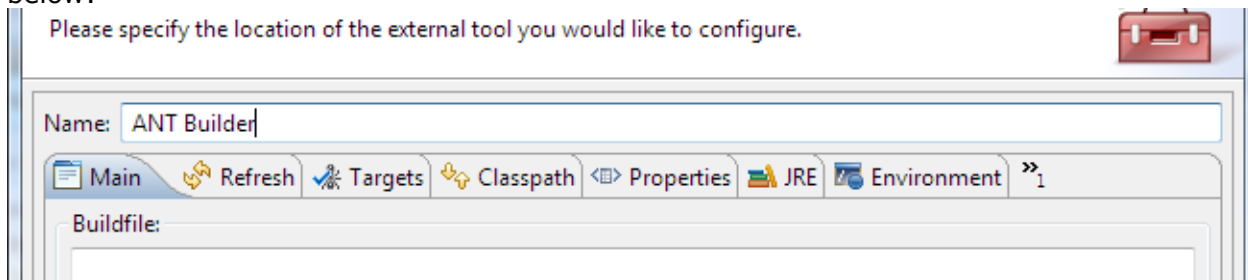
To configure for **ANT builder**, click on **Builders**. Now, deselect the existing **Java Builder** and click on **New** button.



Select **Ant Builder** and then click **OK**.

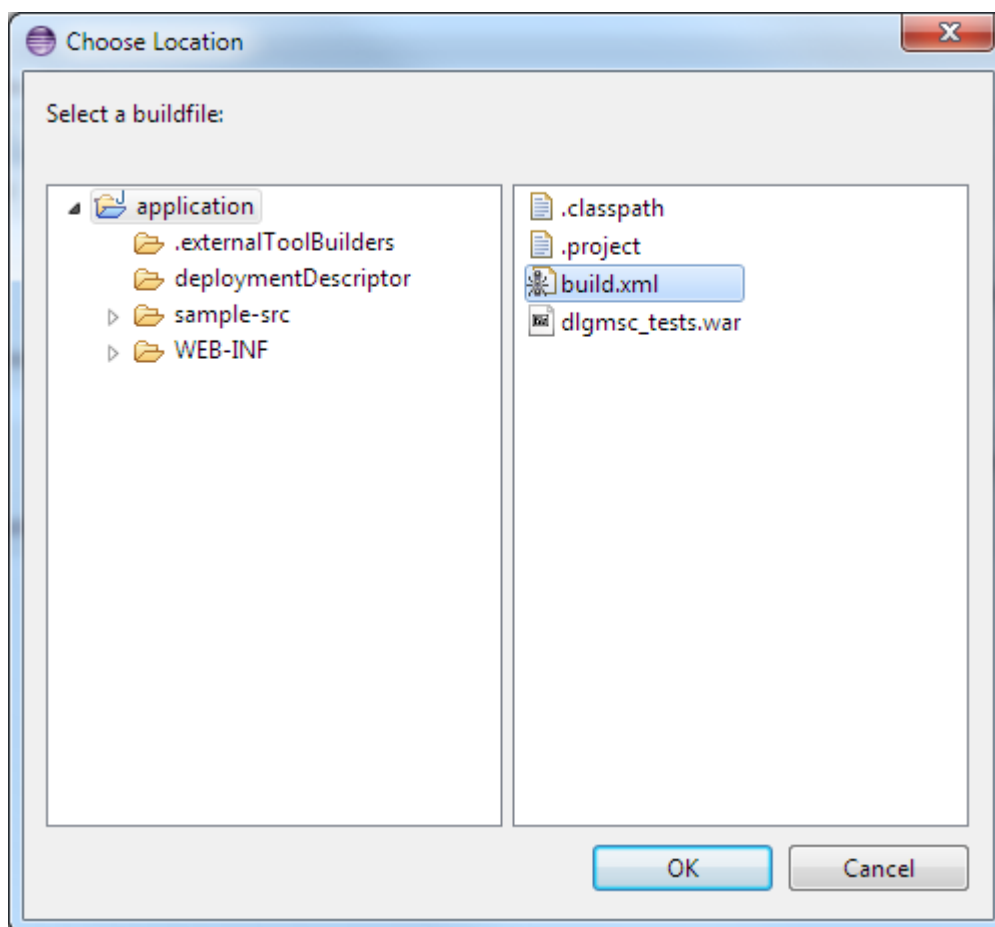


To name the builder, enter in a name in the **Name** section as shown below:

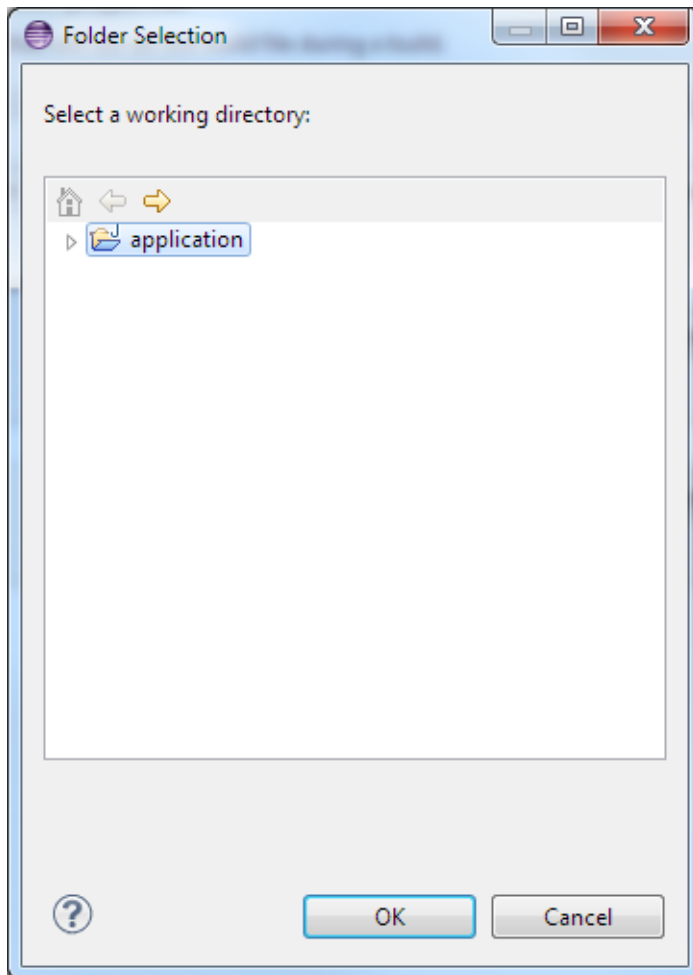


Use the **Main** tab to define **Buildfile** and **Base Directory**:

- Under **Buildfile**, click on **Browse Workspace** button and select the *build.xml* file in the **application** directory. Then, click **OK**:

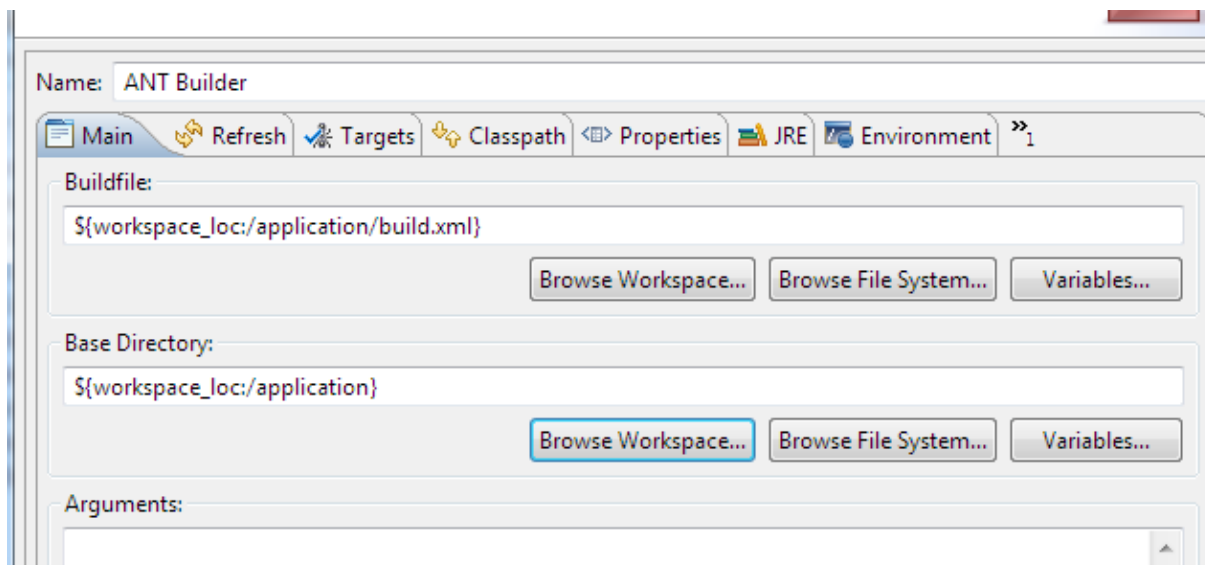


Next, under **Base Directory**, click on **Browse Workspace** button and select the **application** directory.



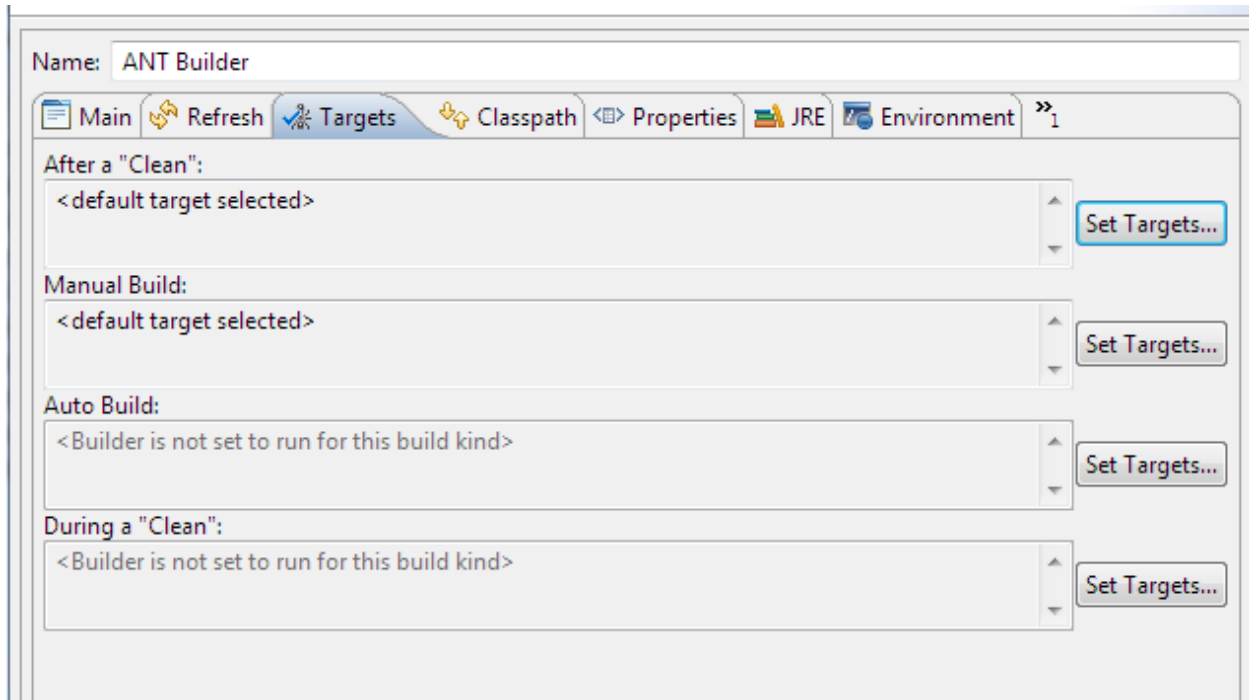
- Then, click **OK**.

The above changes will reflect the main configuration menu as shown below:

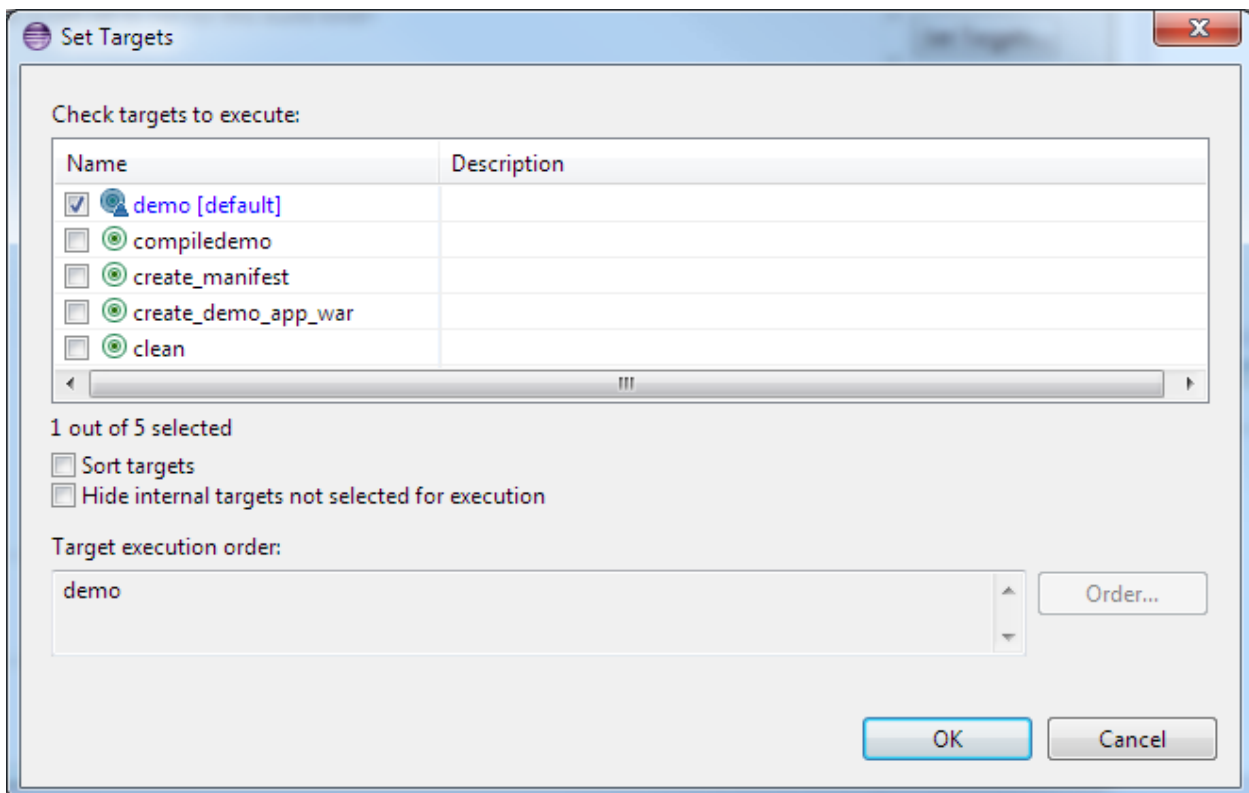




Now, select the **Targets** tab:



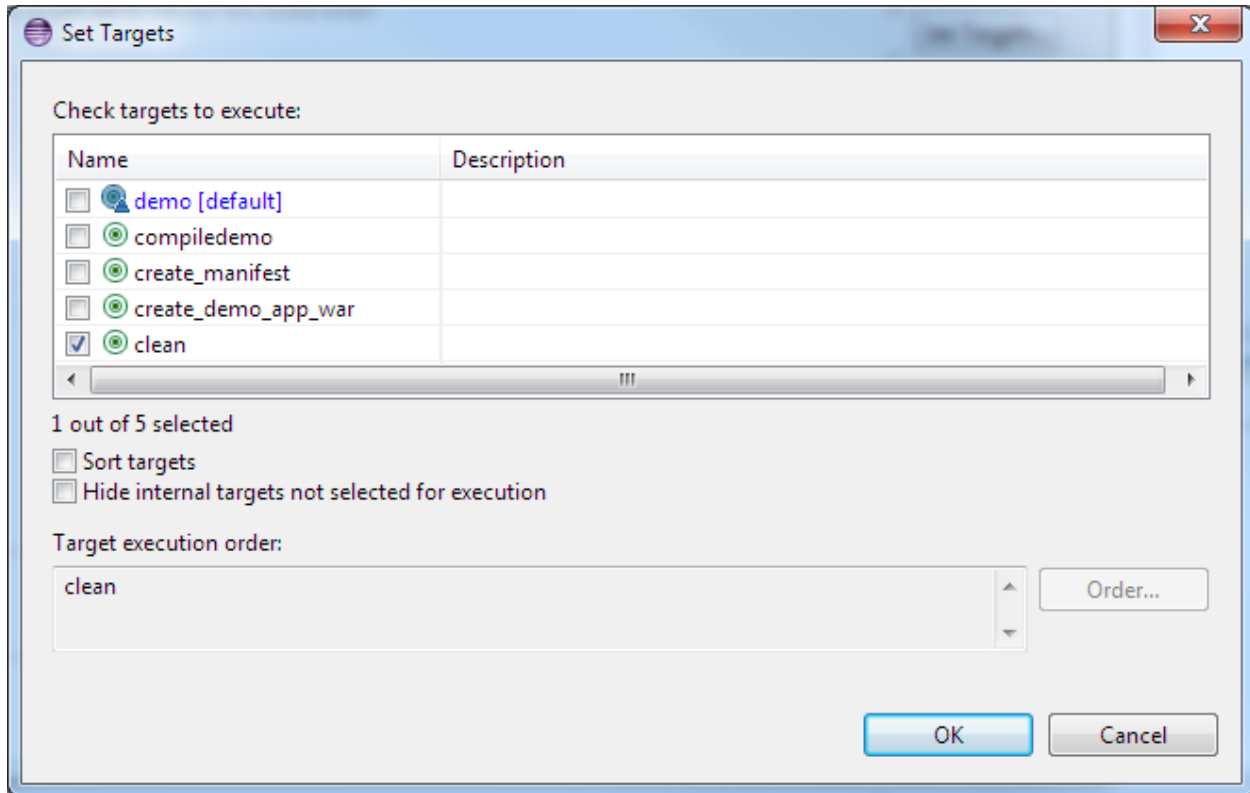
Under **Auto Build**, click the **Set Targets** button. The **demo** target must be selected as illustrated below:



Then, click **OK**.

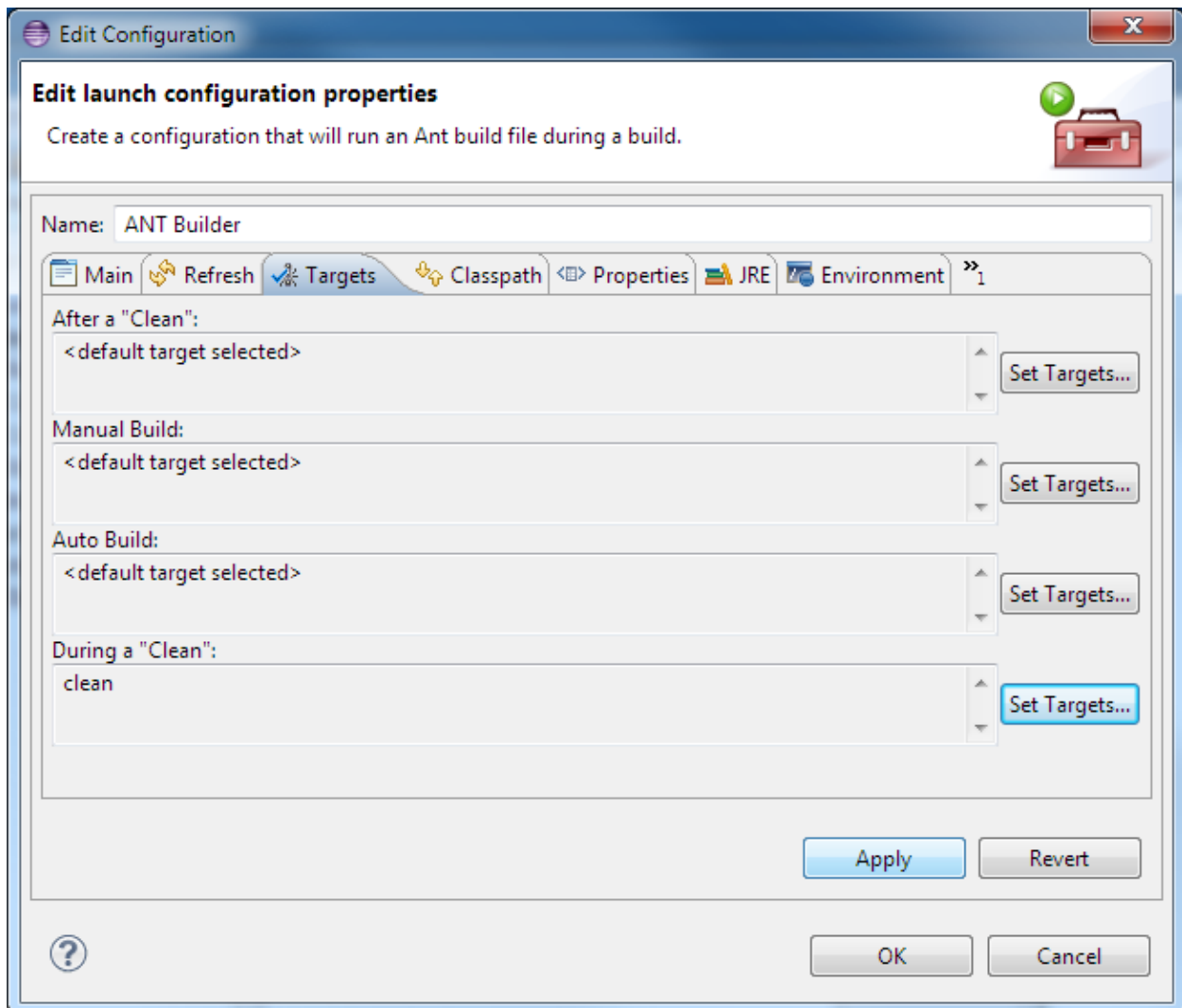
You will now be returned to **Targets** tab on the main configuration menu. Under **During a "Clean"**, click on **Set Targets** button. The only target that should be selected is the **clean** target.

**Note:** The **demo** target will most likely be selected by default, in which case you will need to deselect it.

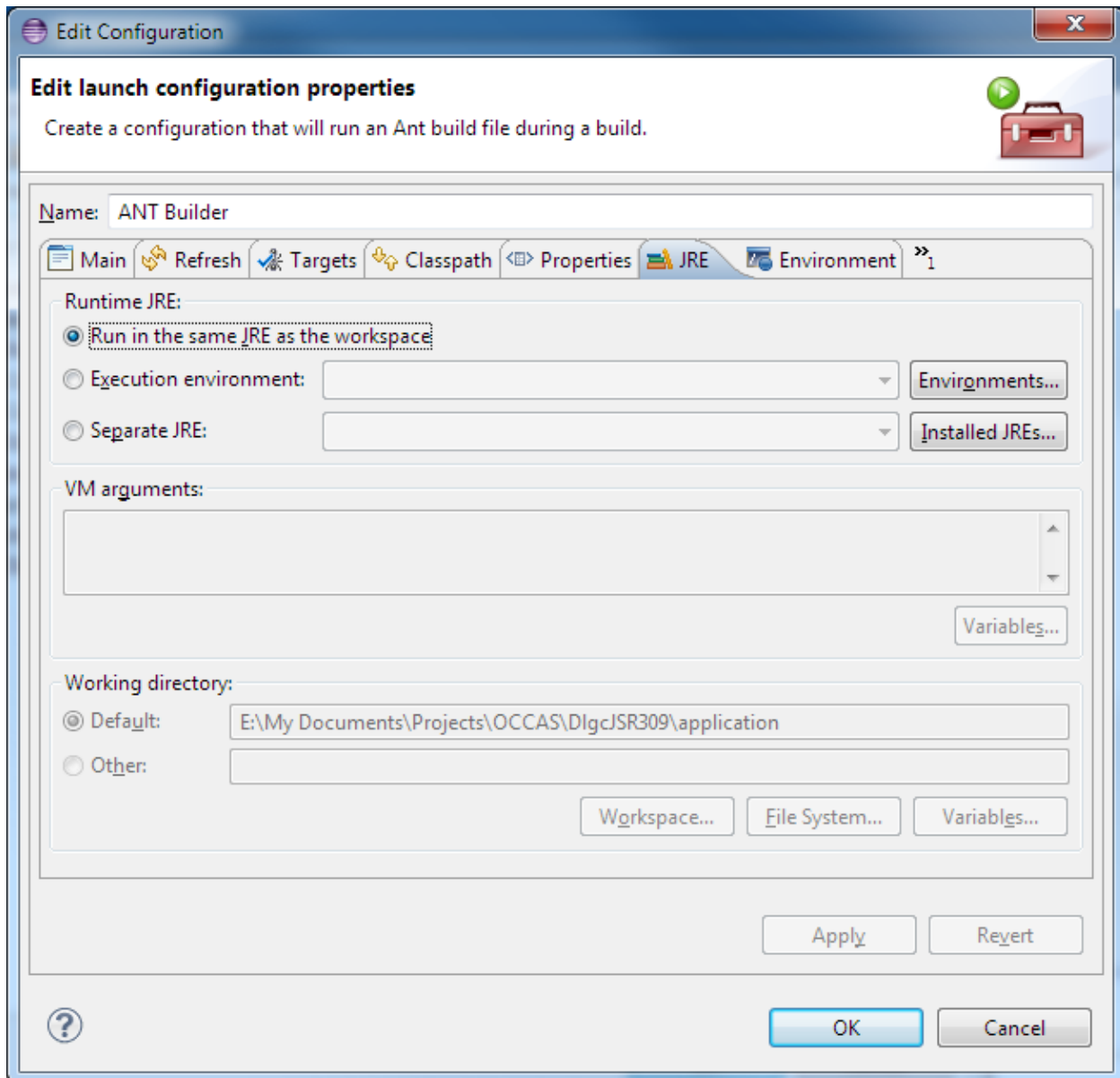


Then, click **OK**.

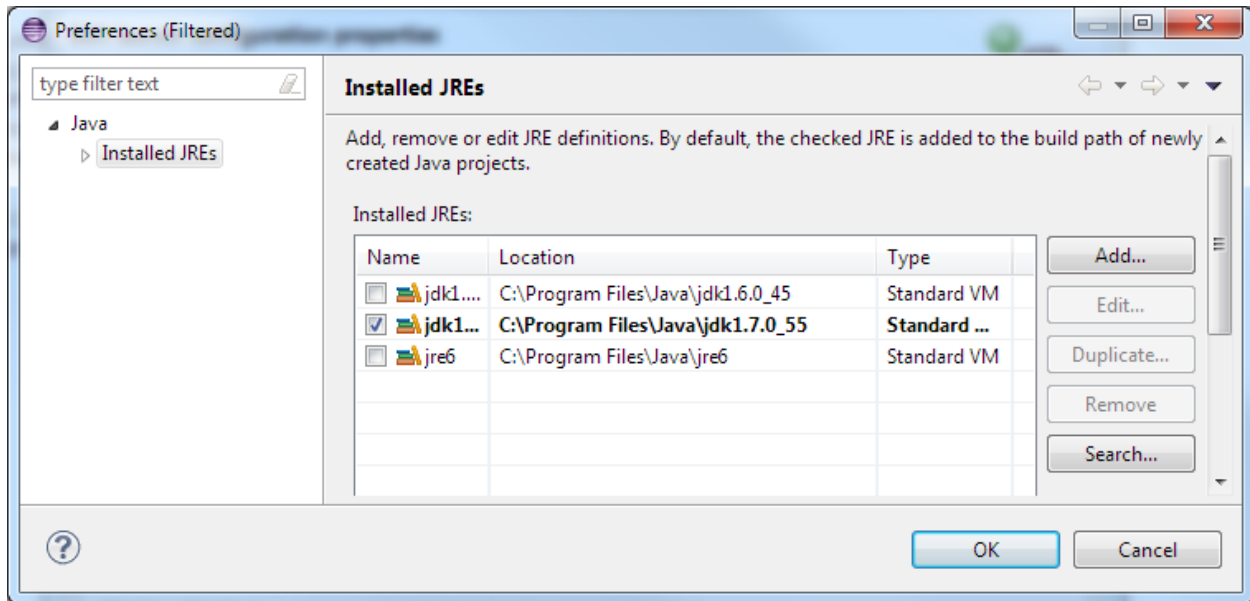
Once returned to **Targets** tab on the main configuration menu, click on **Apply** button:



Next, to select the appropriate **JRE environment** which will be used for this project, click on **JRE** tab:



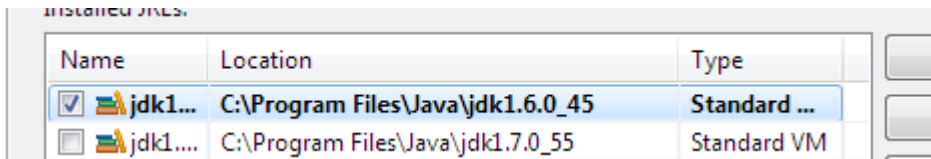
Under **Runtime JRE**, click on **Separate JRE** radio button. Then, click on **Installed JREs** button:



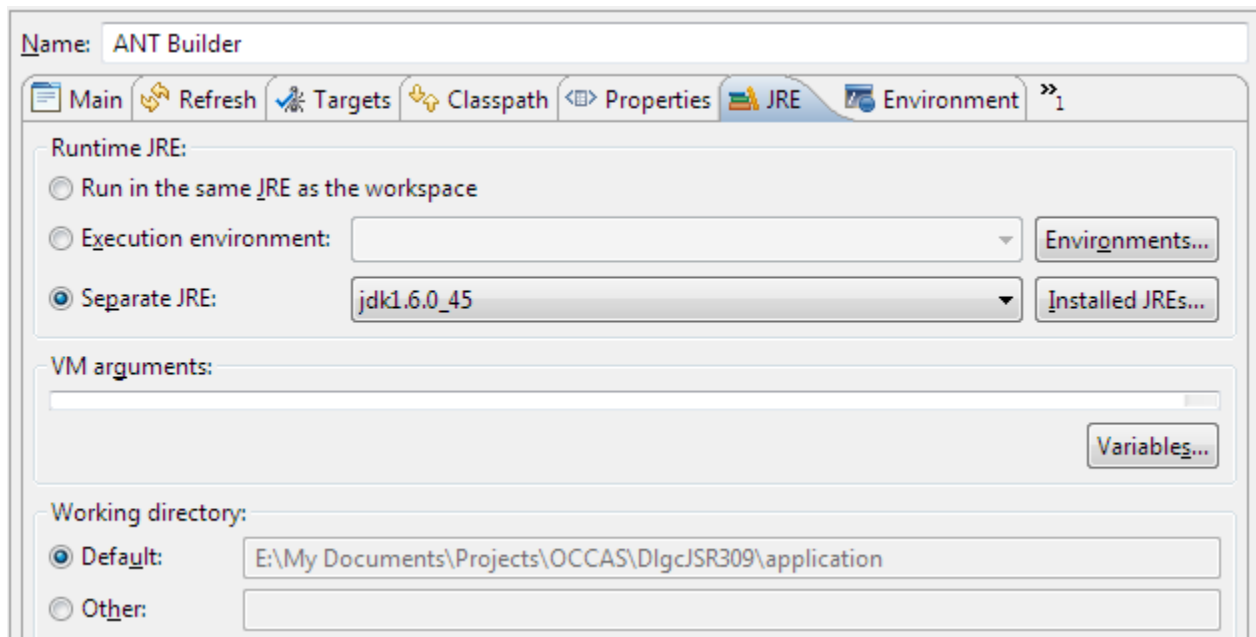
Select the installed *jdk1.6.0\_45* file as shown above.

**Note:** If this version does not show under **Installed JREs**, click **Add**. Then, select **Standard VM** and navigate to the location of your installed *jdk1.6.0\_45* file by clicking on the **Directory** button.

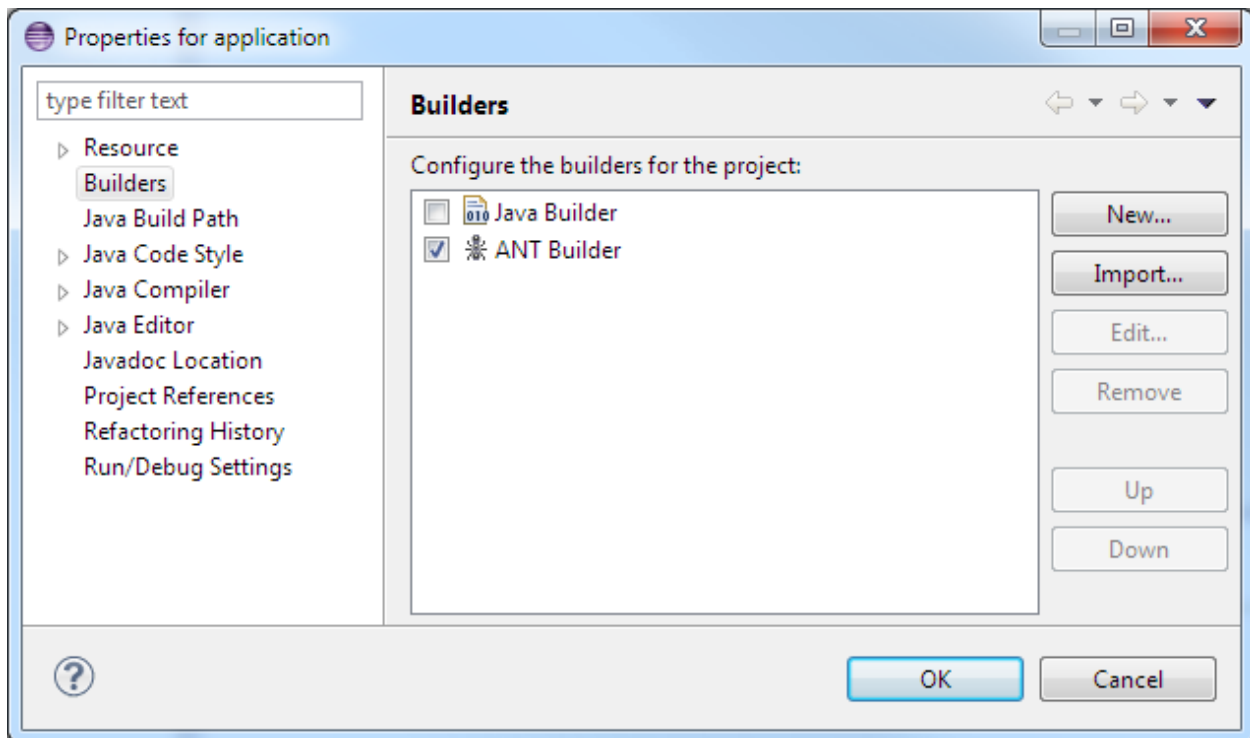
Select the appropriate **JRE** and click **OK**:



Now you have configured the appropriate **JRE** to be used by this project:

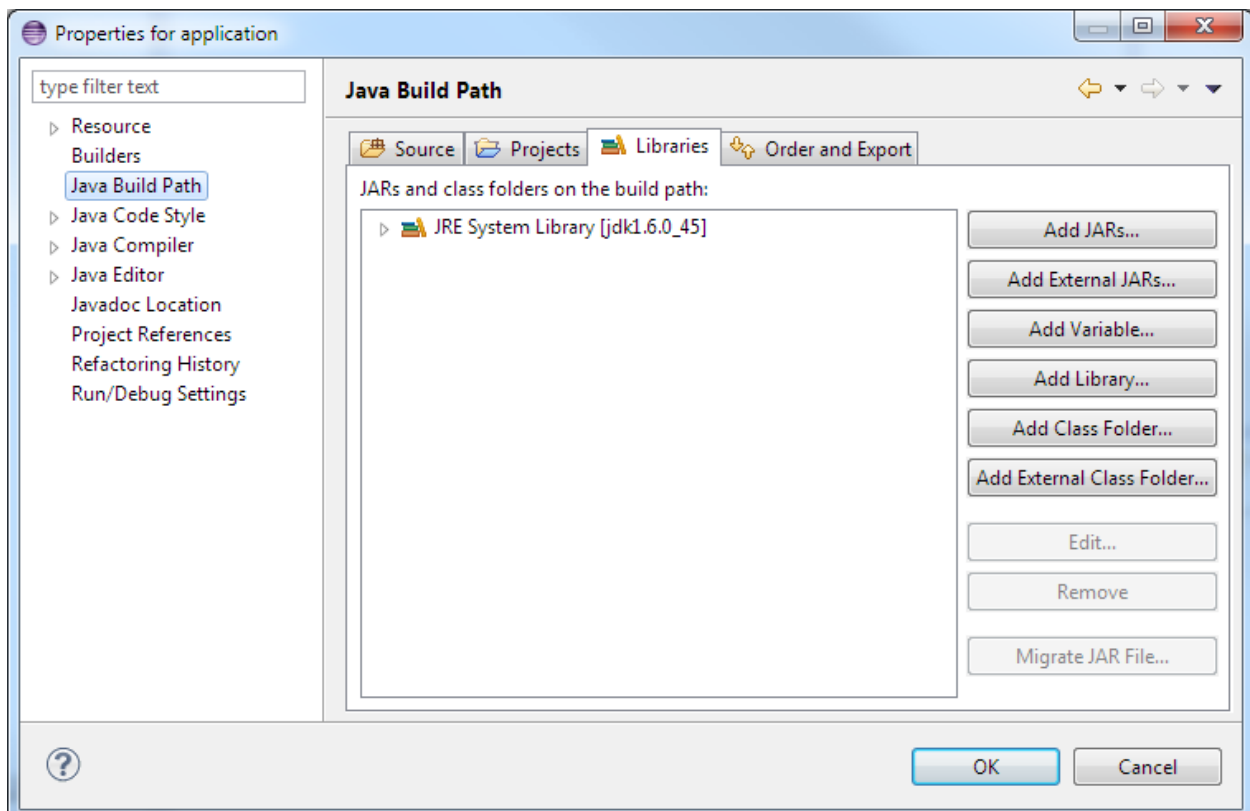


Click **Apply** and then click **OK**.



To ensure that the newly created builder (**ANT Builder**) is at the top of the list, click on **ANT Builder** and position it by clicking on **Up** button.

Next, the **Java Build Path** needs to be configured. Click on **Java Build Path** and then click on **Libraries** tab:



Click on **Add External JARs** button. Locate and click on "*DlgcJSR309/lib*" directory. Select all the files in that directory and click **Open**.

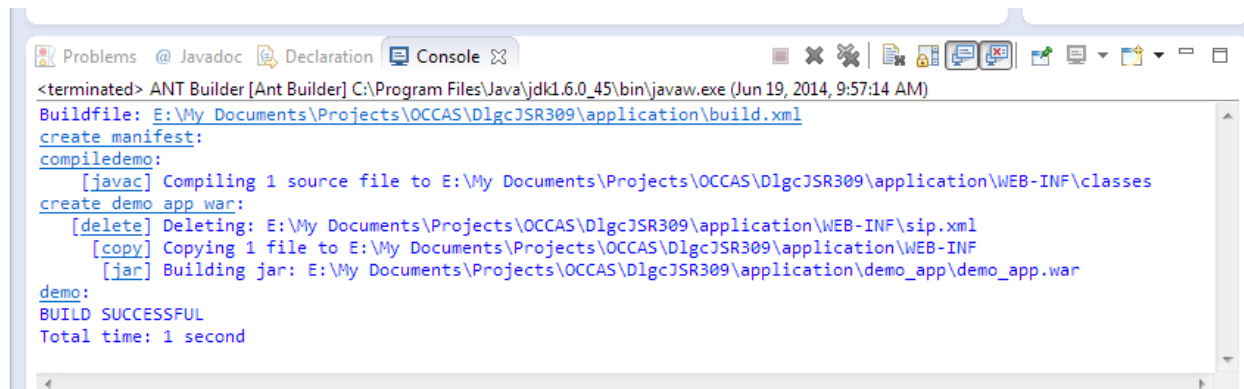
List of JAR files:

- *dlgcsmltypes.jar*
- *geronimo-commonj\_1.1\_spec-1.0.jar*
- *jain-sip-sdp-1.2.91.jar*
- *json\_simple-1.1.jar*
- *log4j-api-2.2.jar*
- *log4j-core-2.2.jar*
- *log4j-slf4j-impl-2.2.jar*
- *mscontrol.jar*
- *msmltypes.jar*
- *org.osgi-3.0.0.jar*
- *slf4j-api-1.7.5.jar*
- *xbean.jar*

Now, click **OK**. The project configuration is now concluded.

## 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 All**. Successful build content will be shown in the **Console** view in Eclipse:



```
<terminated> ANT Builder [Ant Builder] C:\Program Files\Java\jdk1.6.0_45\bin\javaw.exe (Jun 19, 2014, 9:57:14 AM)
Buildfile: E:\My Documents\Projects\OCCAS\DlgcJSR309\application\build.xml
create_manifest:
compiledemo:
  [javac] Compiling 1 source file to E:\My Documents\Projects\OCCAS\DlgcJSR309\application\WEB-INF\classes
create_demo_app_war:
  [delete] Deleting: E:\My Documents\Projects\OCCAS\DlgcJSR309\application\WEB-INF\sip.xml
  [copy] Copying 1 file to E:\My Documents\Projects\OCCAS\DlgcJSR309\application\WEB-INF
  [jar] Building jar: E:\My Documents\Projects\OCCAS\DlgcJSR309\application\demo_app\demo_app.war
demo:
BUILD SUCCESSFUL
Total time: 1 second
```

The newly built application WAR file will be located under the "*DlgcJSR309\application\demo\_app*" directory named *demo\_app.war*. In order to deploy this application, follow the same deployment instructions as described in the [Installation and Configuration of JSR 309 Connector Demo](#).

## 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 5.1.0 for debugging purposes, developers need to follow two simple steps:

- Have Eclipse successfully build the JSR 309 Connector Demo Application WAR file and deploy it in OCCAS 5.1.0.
- Configure OCCAS 5.1.0 for remote debugging.
  - Stop OCCAS 5.1.0.
  - In OCCAS 5.1.0, edit the *startWeblogic.sh* script file and add the following line, enabling the remote debugging, to the startup section as illustrated below:

```
DEBUG_OPTS="-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n"
```

```
# START WEBLOGIC
echo "starting weblogic with Java version:"
DEBUG_OPTS="-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n"
${JAVA_HOME}/bin/java ${JAVA_VM} -version
if [ "${WLS_REDIRECT_LOG}" = "" ]; then
echo "Starting WLS with line:"
echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} `${DEBUG_OPTS}`
`${LOG4J_OPTIONS}` -Dlog4j.debug -Dwlss.local.serialization=${SERIALIZATION_VALUE} -
Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS} ${PROXY_SETTINGS}
`${SERVER_CLASS}`"
`${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} `${DEBUG_OPTS}`
`${LOG4J_OPTIONS}` -Dlog4j.debug -Dwlss.local.serialization=${SERIALIZATION_VALUE} -
Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${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} `${DEBUG_OPTS}`
`${LOG4J_OPTIONS}` -Dlog4j.debug -Dwlss.local.serialization=${SERIALIZATION_VALUE} -
Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS} ${PROXY_SETTINGS}
`${SERVER_CLASS}` > "${WLS_REDIRECT_LOG}" 2>&1
fi
stopAll
```

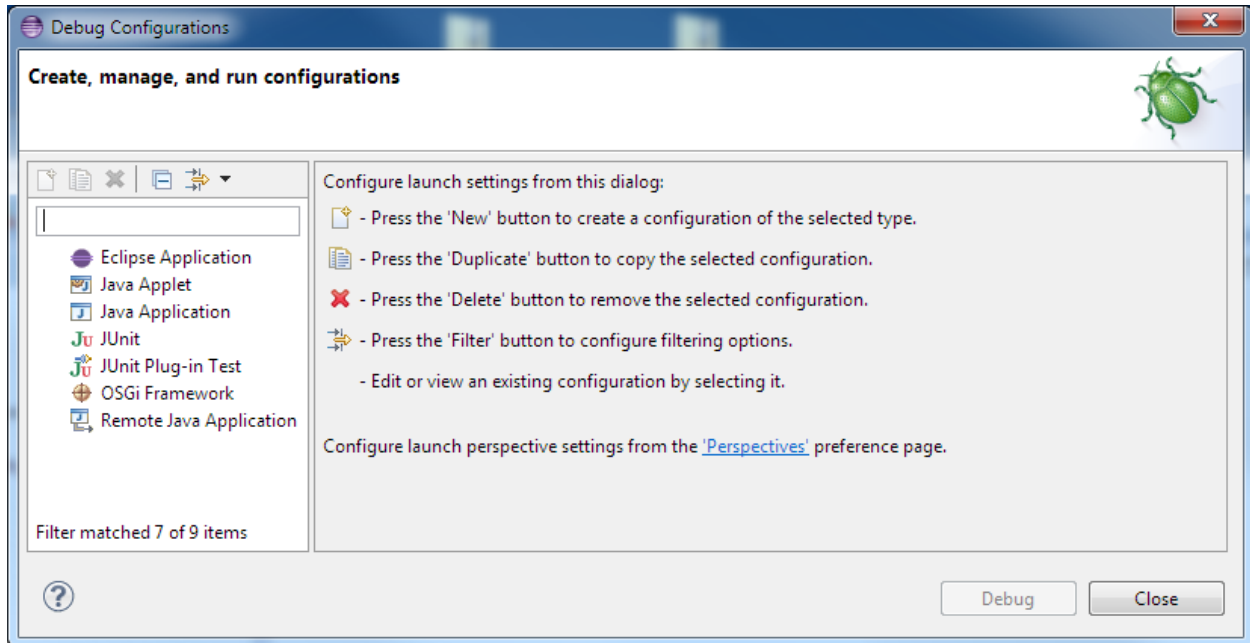
**Note:** The socket address specified above is 8000 but any port of choice can be used. Any port used needs to be enabled in a firewall in order to allow communication through it.

- Start OCCAS 5.1.0 and make sure there are no errors in the console.

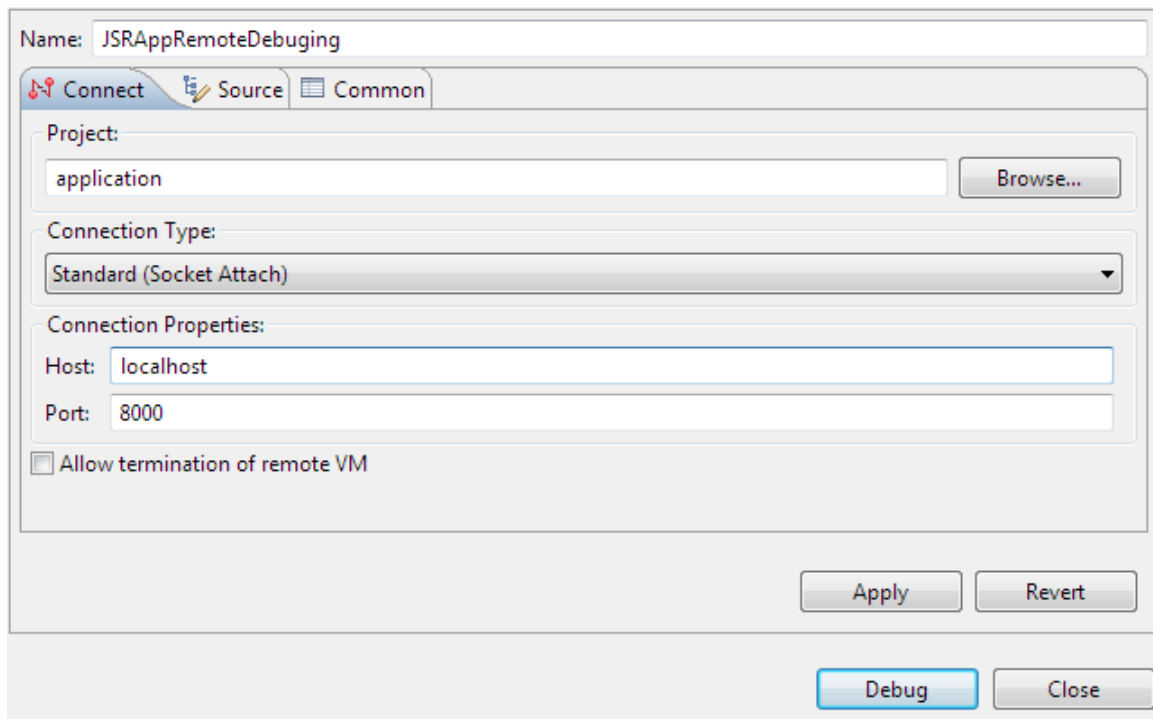


## Eclipse Project Remote Debugging Configuration

When in Eclipse with an active 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, go to the **Run** menu and click on **Debug Configurations**:



Double click on **Remote Java Application**.

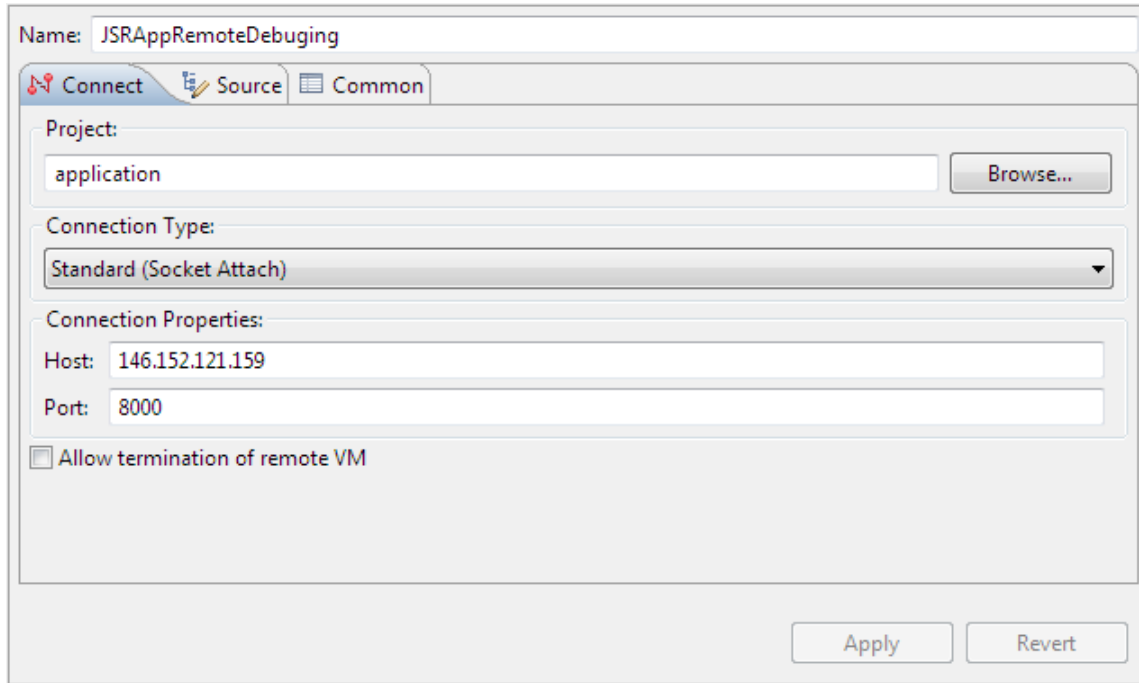


Specify the **Name** for this remote debugging configuration (for example, JSRAppRemoteDebugging).

Under **Connection Properties**, specify the **Host** address, which is the IP address of OCCAS with deployed application, to be debugged.

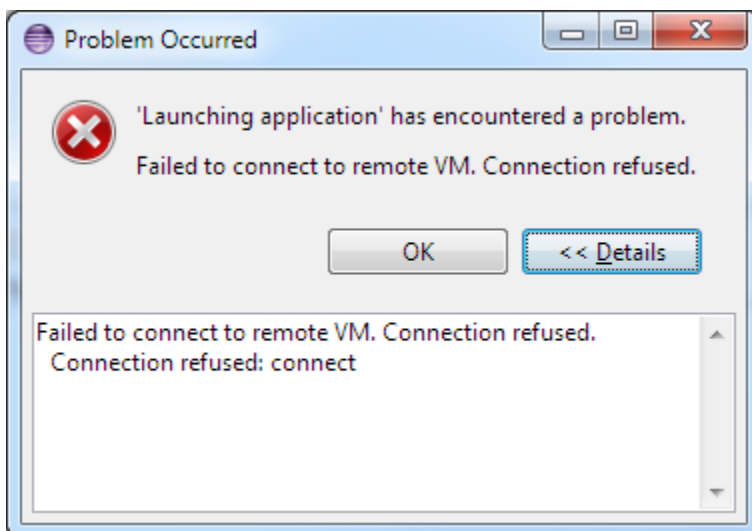
Also, specify the remote debug **Port** to be used for communication with OCCAS as defined in the previous section.

Below is an example of what your window would look like once information is added:



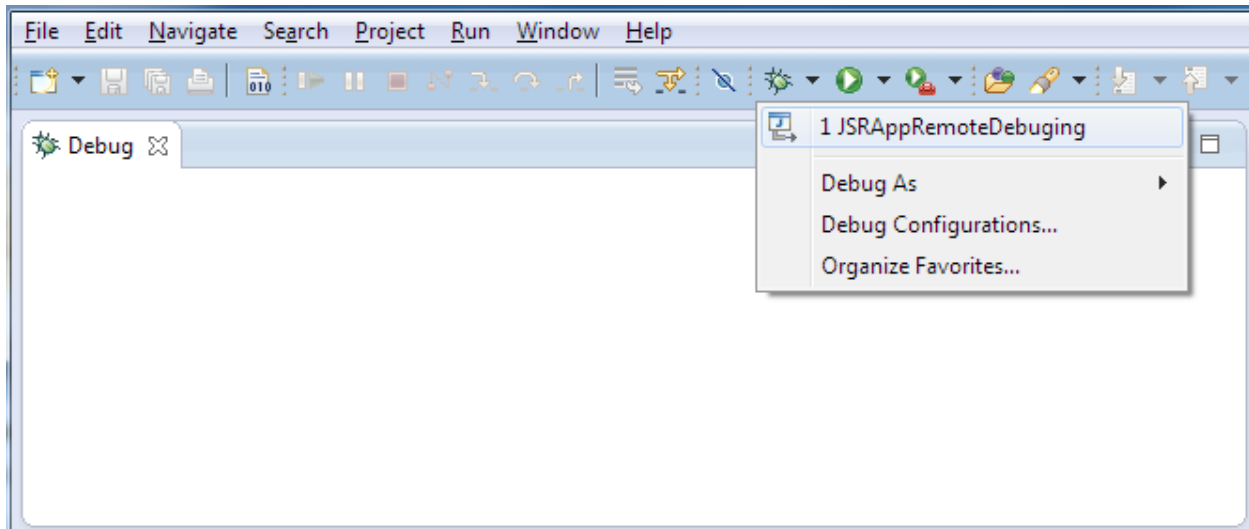
Once done, click **Apply** and then click **Debug**.

OCCAS 5.1.0 needs to be running at this point. If not, Eclipse will report a connection error message. If OCCAS is running but Eclipse is still reporting a connection error, this could be due to either a port mismatch between Eclipse and what OCCAS was configured for or because of firewall settings on OCCAS not allowing the specified port.

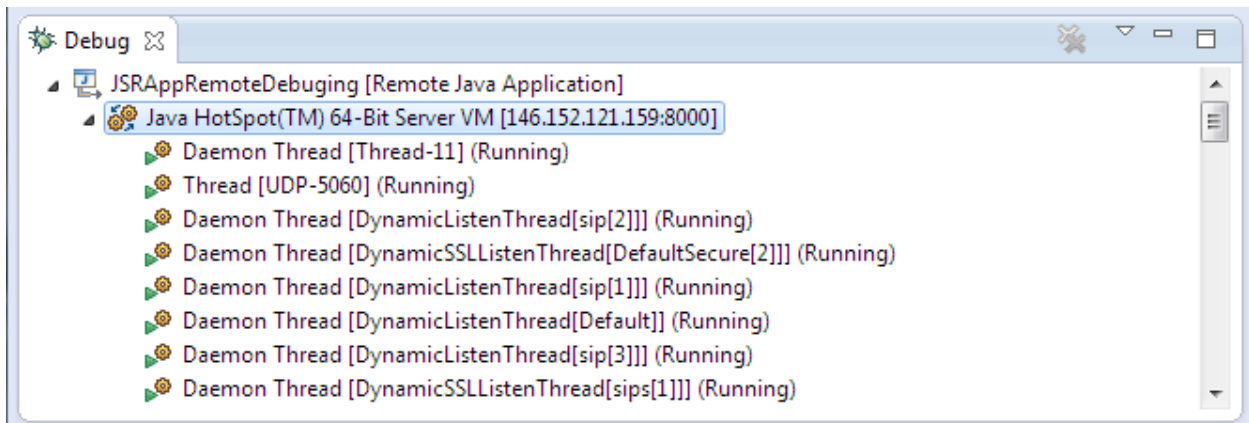


Now, open **debug perspective** in Eclipse (**Windows > Open Perspective > Debug**).

If nothing shows under the **Debug** section of a **debug perspective**, then a connection to OCCAS has not been established. To connect/reconnect, go to the **debug icon** on the **toolbar** and choose the newly created **remote debugging configuration**:



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 5.1.0.

## 7. Appendix A: JSR 309 Connector Environment Setup

---

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

- [Installing and Configuring the OCCAS](#)

For system requirements and supported platforms, see [JSR 309 Connector Requirements](#).

This section does not go into details of OCCAS, but will help build an OCCAS system which could be used for verification purposes.

Steps to complete on OS level include:

- Enable NTP (Network Time Protocol)
- Enable ports in 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.

If you need more details on OCCAS, refer to the OCCAS installation instructions available from [www.oracle.com](http://www.oracle.com).

### Installing and Configuring the OCCAS

This section describes the installation and configuration instructions for OCCAS 5.1.0. This section illustrates how to install and configure OCCAS in order to be able to go to the next step of [Installing the JSR 309 Connector](#).

**Note:** If you are familiar with OCCAS or are planning to deploy on an existing OCCAS setup, proceed to [Installing the JSR 309 Connector](#).

Here are some highlights of the necessary steps:

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

### Pre-Installation Setup

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.

Run the following command at the prompt:

```
service network restart
```

## JDK Setup

Download the latest JDK *.rpm* file from [www.oracle.com](http://www.oracle.com). For example, the following setup is based on *jdk-6u45-linux-x64.rpm* file.

Install the JDK *.rpm* file:

```
rpm -ivh jdk-6u45-linux-x64.rpm
```

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

```
Export JAVA_HOME=/usr/java/jdk1.6.0_45
```

Save the *.bashrc* file and then execute:

```
source ~/.bashrc
```

This will take the changes into effect on the system.

Move the *occas5.1.0.zip* file into the root directory.

Unzip the *occas5.1.0.zip* file then proceed to [OCCAS Installation](#).

## OCCAS Installation

Install the *occas510\_ja\_generic.jar* file.

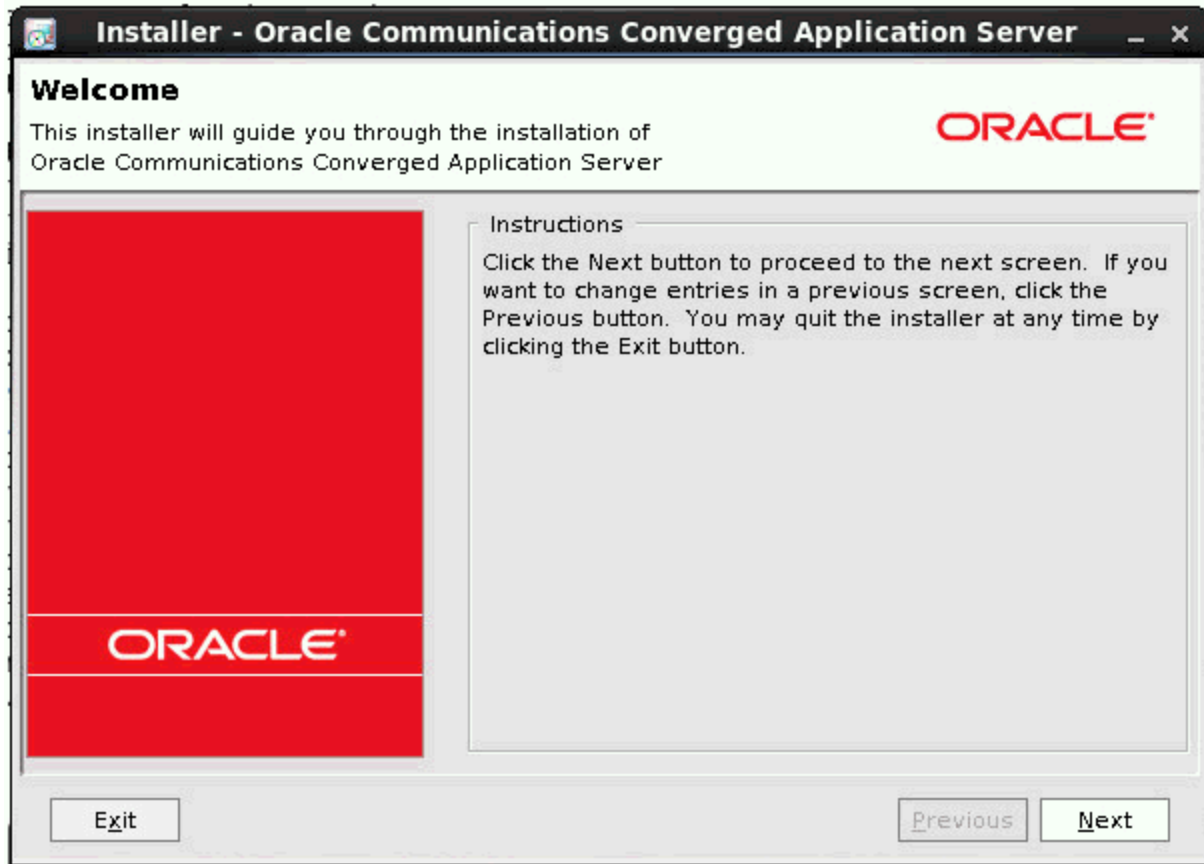
```
java -d64 -jar occas510_ja_generic.jar
```

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

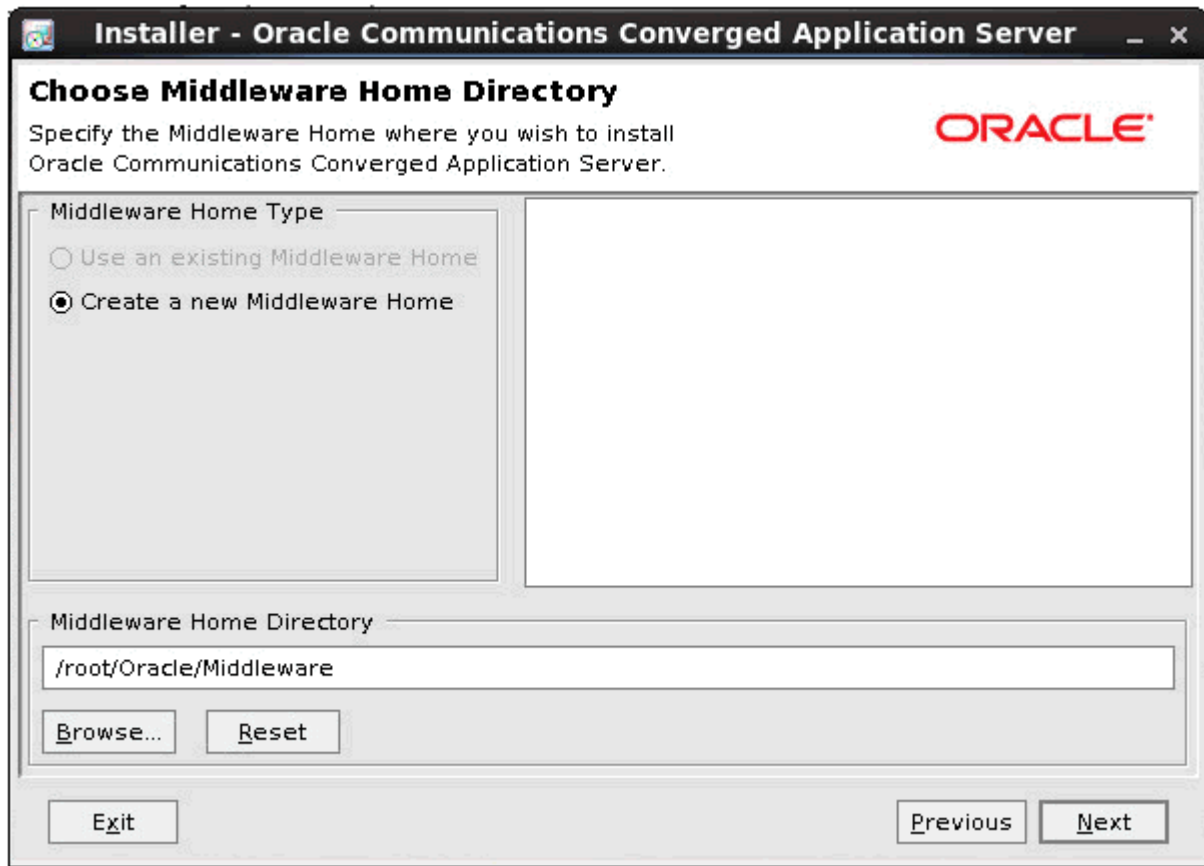
```
root@occas5:~
File Edit View Search Terminal Help
[root@occas5 ~]# vim /etc/hosts
[root@occas5 ~]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)      [ OK ]
Shutting down loopback interface:                                [ OK ]
Bringing up loopback interface:                                  [ OK ]
Bringing up interface eth0: Active connection state: activated
Active connection path: /org/freedesktop/NetworkManager/ActiveConnection/1 [ OK ]

[root@occas5 ~]# ls
anaconda-ks.cfg  Downloads      Music          Public
Desktop          install.log    occas5.1.0.zip Templates
Documents        install.log.syslog Pictures        Videos

[root@occas5 ~]# unzip occas5.1.0.zip
Archive: occas5.1.0.zip
  inflating: occas510_ja_generic.jar
[root@occas5 ~]# java -d64 -jar occas510_ja_generic.jar
[root@occas5 ~]# java -d64 -jar occas510_ja_generic.jar
Extracting 0%.....
```



Click on **Next**.



Select **Create a new Middleware Home**, then click on **Next**.

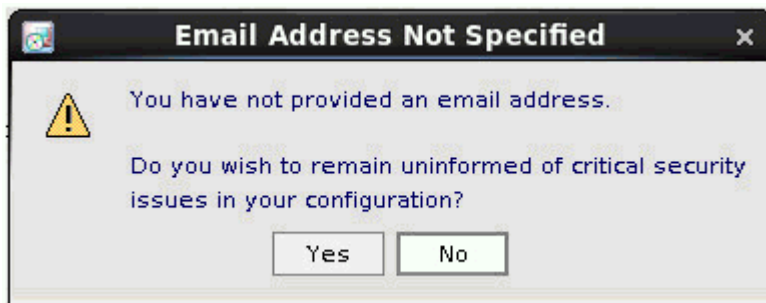


Deselect **security updates** (unless you have an account with Oracle), then click on **Next**.

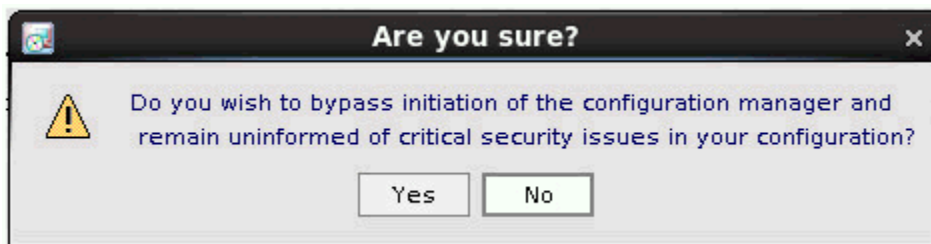




Click on **Next**.



Click on **Yes**.



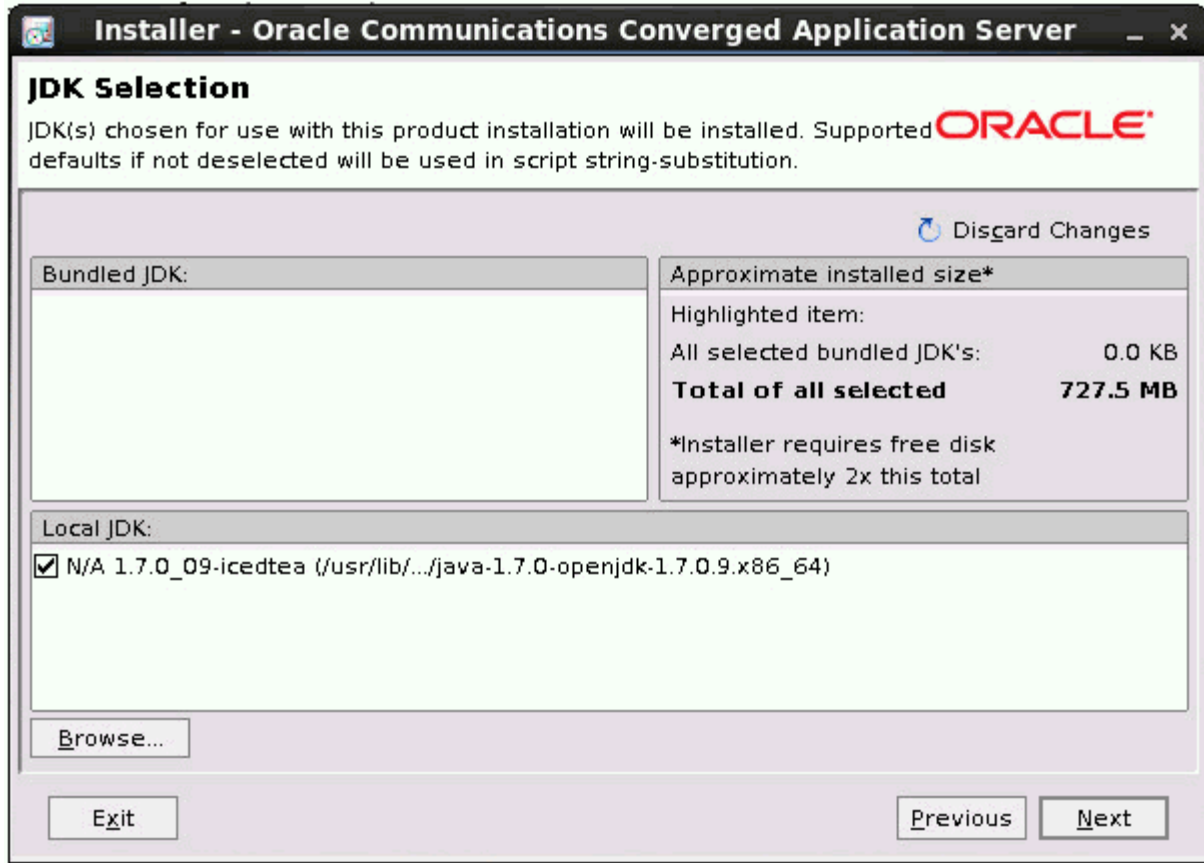
Click on **Yes**.



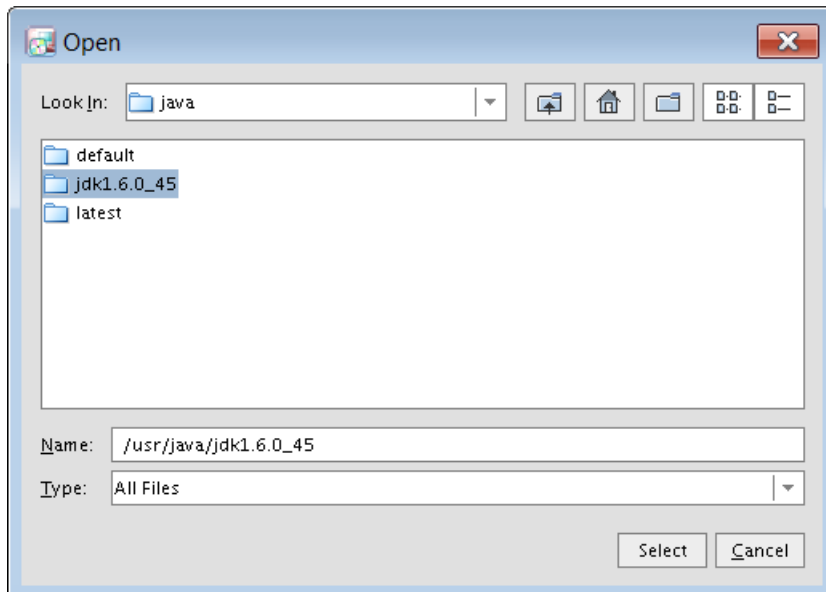
Select **I wish to remain uninformed**, then click **Continue**.



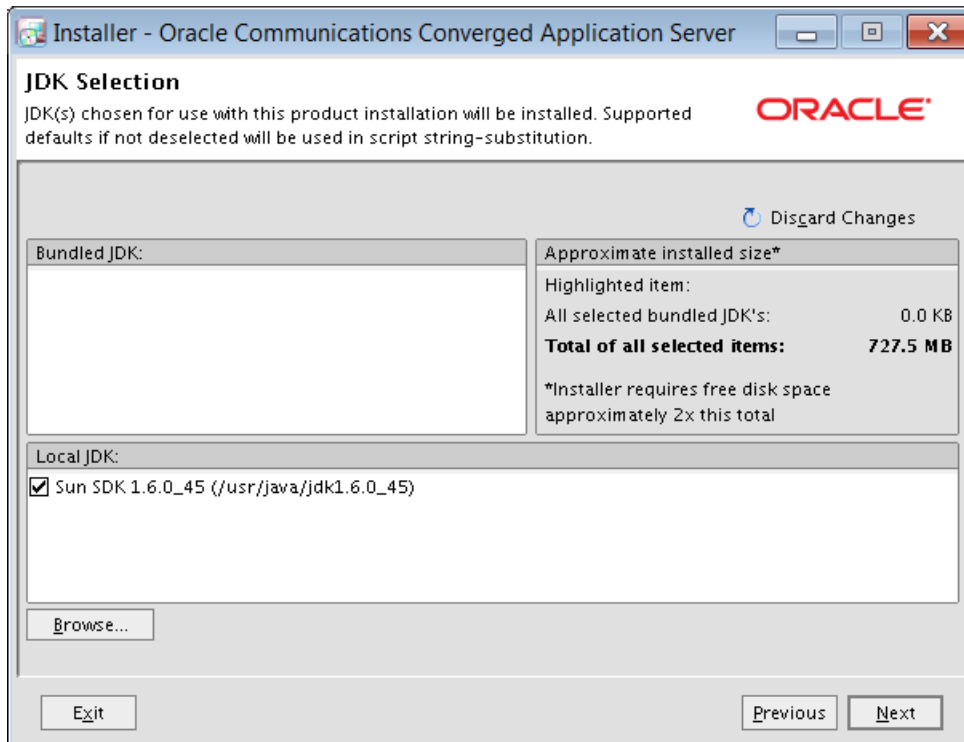
Select **Typical**, then click on **Next**.



Browse to the previously installed **jdk1.6.0\_45** directory.



Select **jdk1.6.0\_45** directory then click on **Select**.



Make sure the selected **jdk1.6.0\_45** directory is the only JDK checked. Then, click on **Next**.



Click on **Next**.



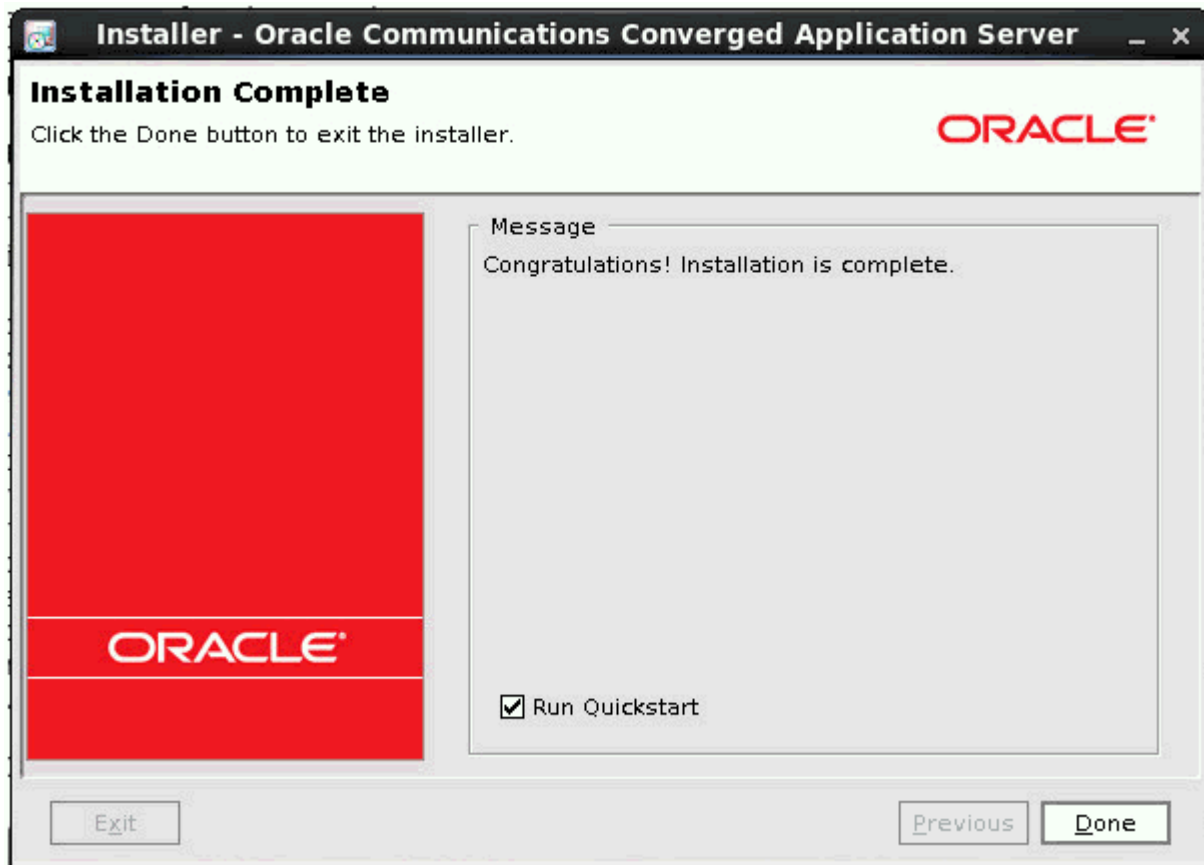
Click on **Next**.



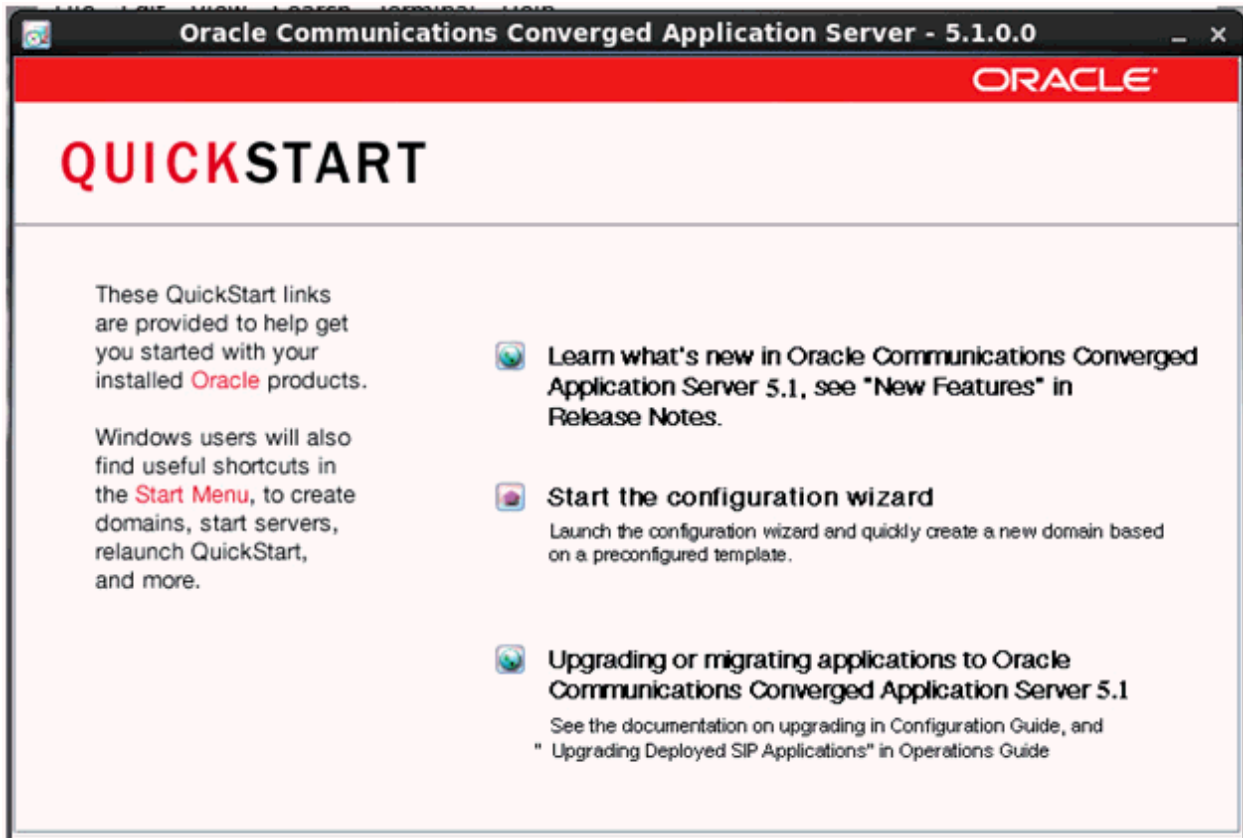
Click on **Next**.

At this point, OCCAS is installed. The steps in the next section will go over [OCCAS Configuration](#).

## OCCAS Configuration

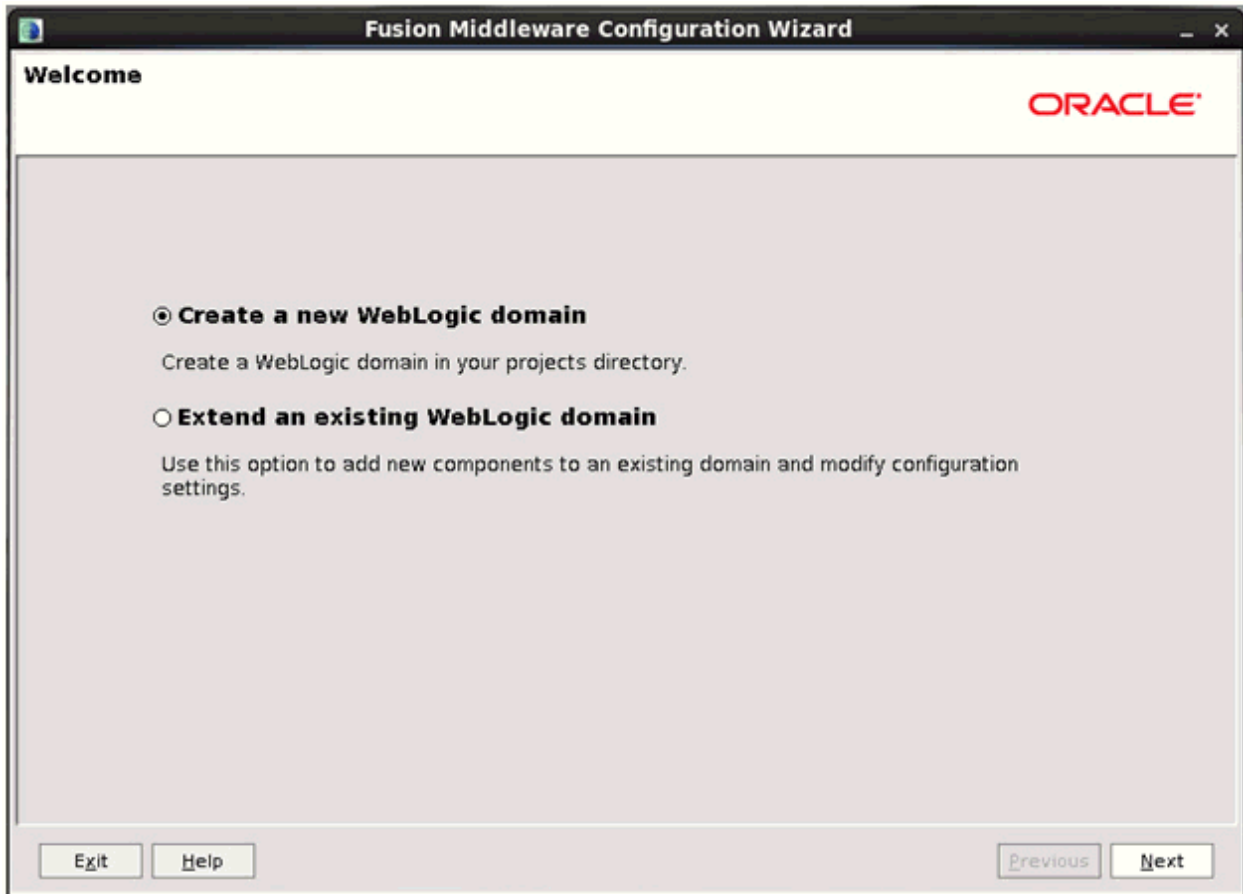


Select **Run Quickstart**, then click on **Done**.

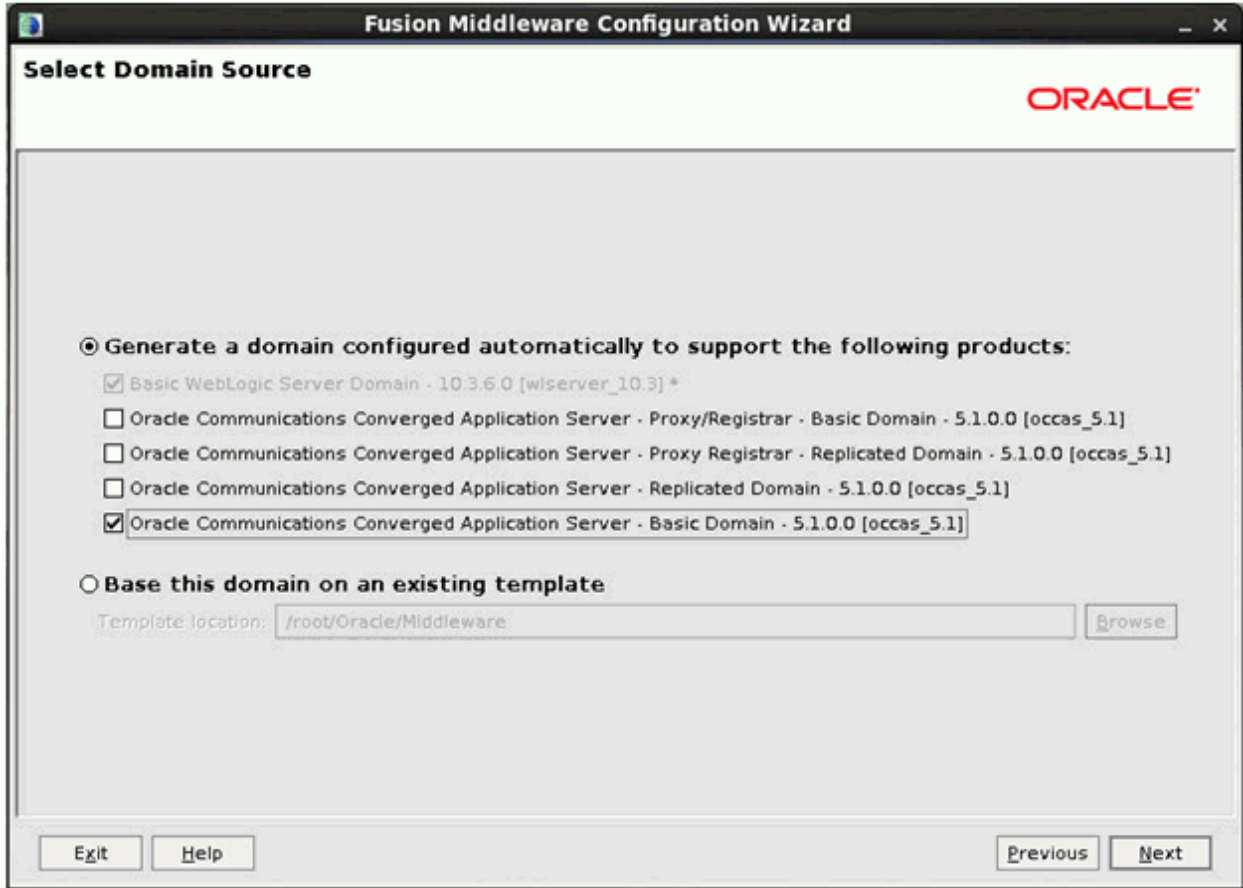


Click on **Start the configuration wizard**.





Select **Create a new WebLogic domain**, then click on **Next**.



Select **Generate a domain configured automatically to support the following products** and choose **Oracle Communications Converged Application Server - Basic Domain - 5.1.0.0 (occas\_5.1)**. Then, click on **Next**.



Click on **Next**.

The screenshot shows a window titled "Fusion Middleware Configuration Wizard" with the subtitle "Configure Administrator User Name and Password". The Oracle logo is in the top right corner. A "Discard Changes" link is in the top left. The main area contains four input fields: "\*Name:" with the text "weblogic", "\*User password:" with masked characters, "\*Confirm user password:" with masked characters, and "Description:" with the text "This user is the default administrator.". At the bottom, there are buttons for "Exit", "Help", "Previous", and "Next".

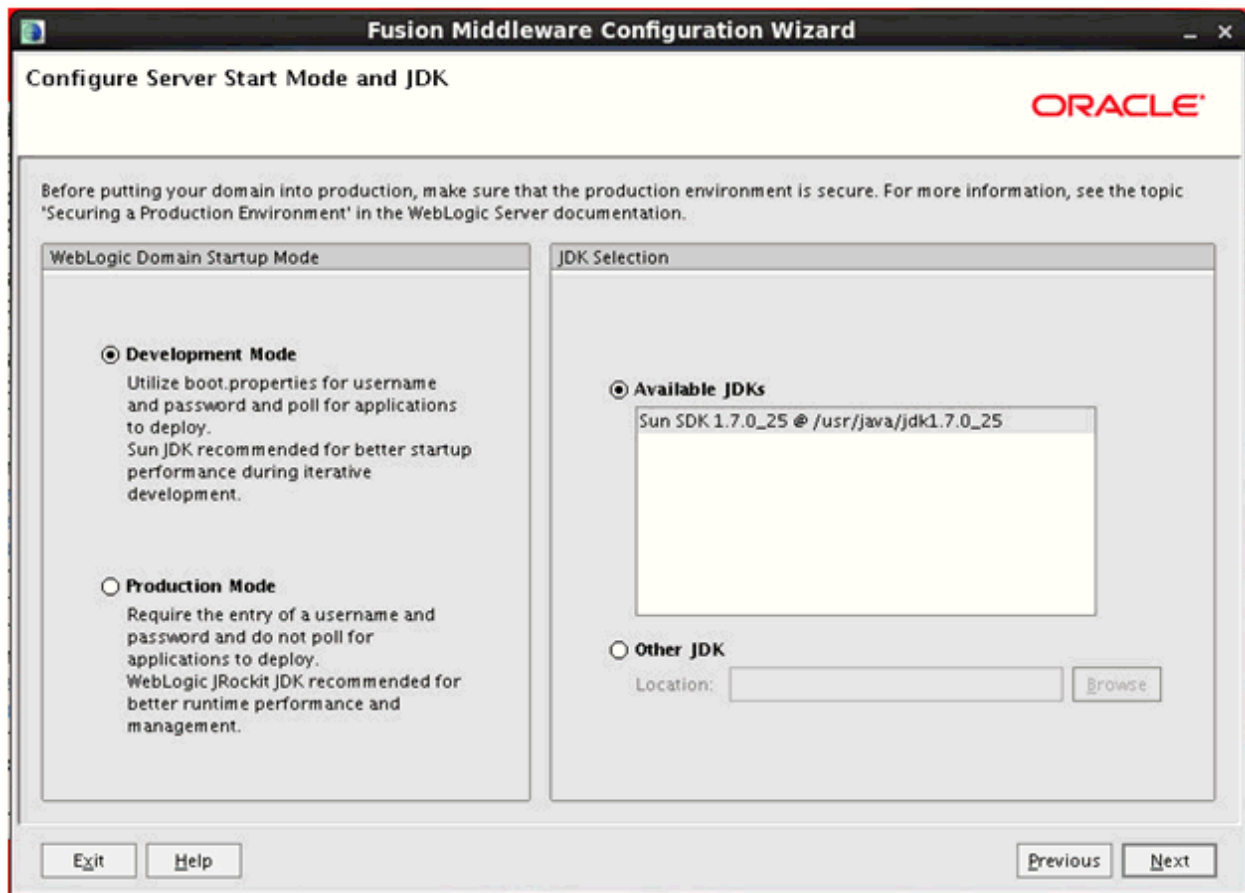
Specify **Name** and **User password**, then click on **Next**.

The following is used as an example:

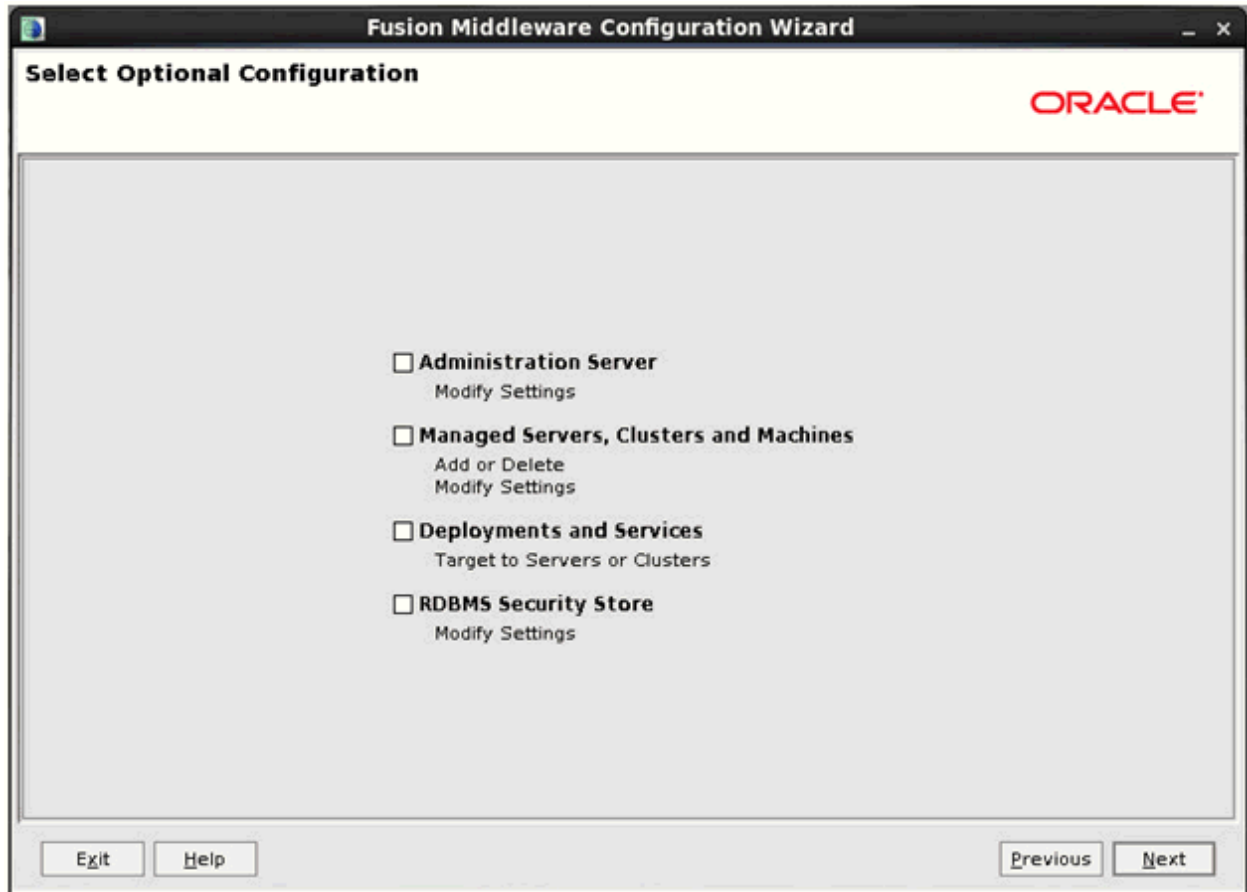
**Name:** weblogic

**User password:** Webl0gic!! ("0" is a zero)

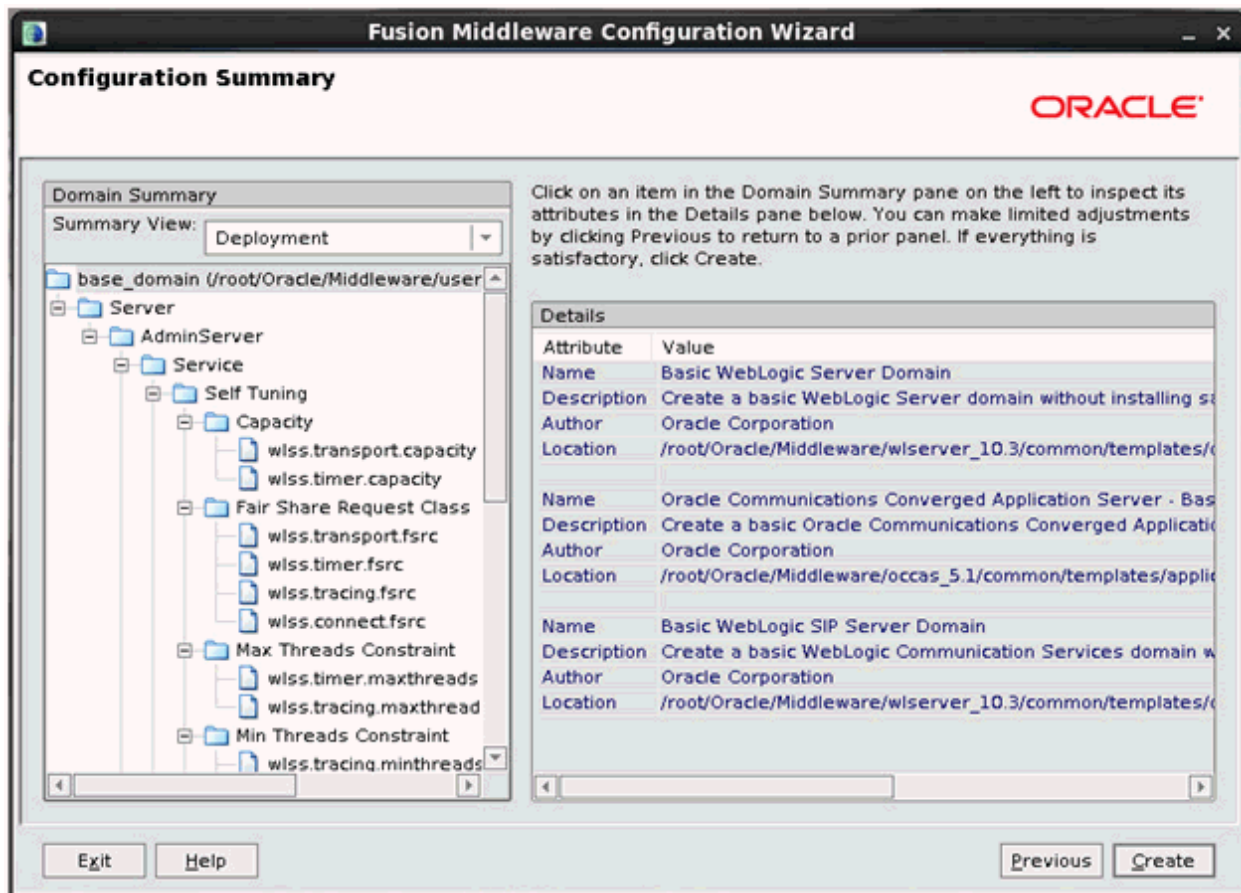
**Note:** A strong password is required.



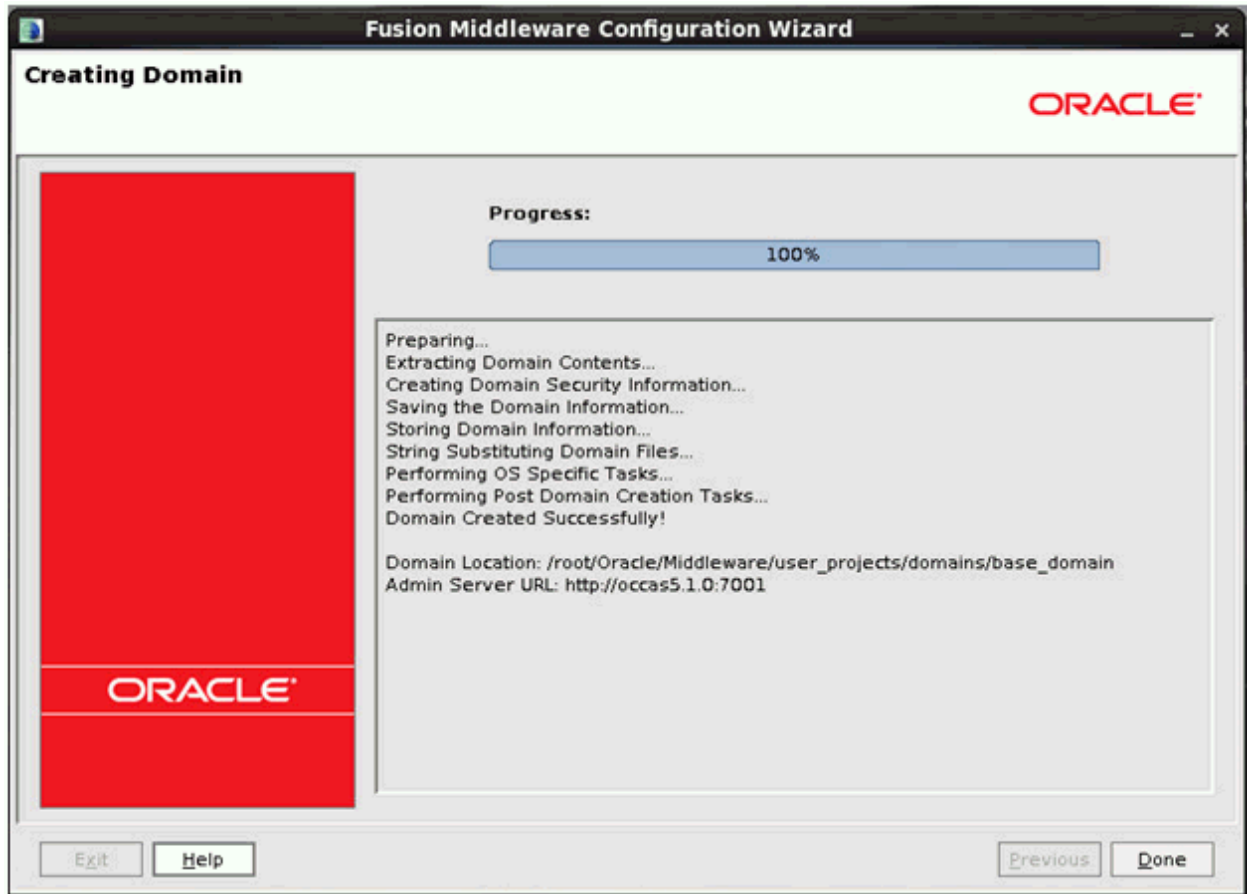
Select **Development Mode**, then click on **Next**.



Click on **Next**.



Click on **Create**.



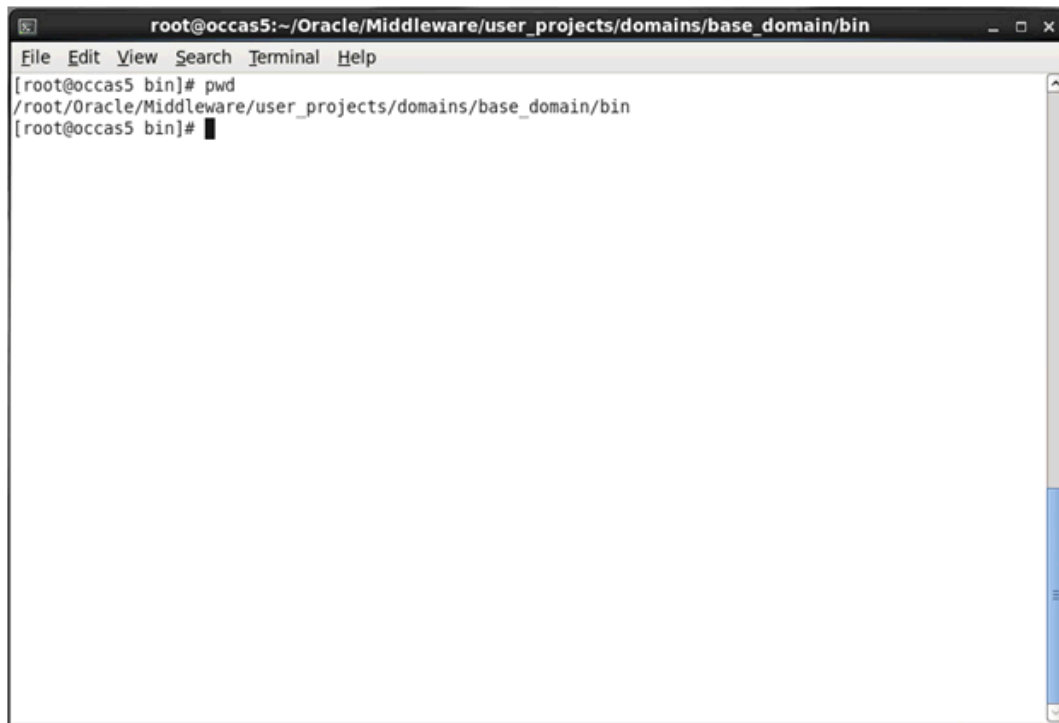
The OCCAS installation and configuration is now complete. Click on **Done**.



## OCCAS Startup

To start OCCAS, go to the "<Domain Location>/bin" directory:

```
/root/Oracle/Middleware/user_projects/domains/base_domain/bin
```



```
root@occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
File Edit View Search Terminal Help
[root@occas5 bin]# pwd
/root/Oracle/Middleware/user_projects/domains/base_domain/bin
[root@occas5 bin]#
```

Run the following command:

```
./startWebLogic.sh
```



```
root@occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
File Edit View Search Terminal Help
[root@occas5 bin]# pwd
/root/Oracle/Middleware/user_projects/domains/base_domain/bin
[root@occas5 bin]# ./startWebLogic.sh
```

```
root@occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
File Edit View Search Terminal Help
java version "1.7.0_09-icedtea"
OpenJDK Runtime Environment (rhel-2.3.4.1.0.1.el6_3-x86_64)
OpenJDK 64-Bit Server VM (build 23.2-b09, mixed mode)
Starting WLS with line:
/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.9.x86_64/bin/java -Xms512m -Xmx512m -Dweblogic.Name=AdminServer -Djava.security.policy=/root/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.policy -Dweblogic.ProductionModeEnabled=true -Dwlss.maddr.enable=true -da -Dplatform.home=/root/Oracle/Middleware/wlserver_10.3 -Dwls.home=/root/Oracle/Middleware/wlserver_10.3/server -Dweblogic.home=/root/Oracle/Middleware/wlserver_10.3/server -Dweblogic.management.discover=true -Dwlw.iterativeDev=false -Dwlw.testConsole=false -Dwlw.logErrorsToConsole=false -Dweblogic.ext.dirs=/root/Oracle/Middleware/patch_wls1036/profiles/default/sysexManifest_classpath:/root/Oracle/Middleware/patch_occas510/profiles/default/sysexManifest_classpath weblogic.Server
<Jun 4, 2013 6:58:29 PM EDT> <Info> <Security> <BEA-090905> <Disabling CryptoJ JCE Provider self-integrity check for better startup performance. To enable this check, specify -Dweblogic.security.allowCryptoJDefaultJCEVerification=true>
<Jun 4, 2013 6:58:29 PM EDT> <Info> <Security> <BEA-090906> <Changing the default Random Number Generator in RSA CryptoJ from ECDRBG to FIPS186PRNG. To disable this change, specify -Dweblogic.security.allowCryptoJDefaultPRNG=true>
<Jun 4, 2013 6:58:29 PM EDT> <Notice> <WebLogicServer> <BEA-000395> <Following extensions directory contents added to the end of the classpath:
/root/Oracle/Middleware/user_projects/domains/base_domain/lib/sipactivator.jar>
<Jun 4, 2013 6:58:30 PM EDT> <Info> <Server> <BEA-002647> <The service plugin, com.oracle.core.sip.activator, was added from /root/Oracle/Middleware/user_projects/domains/base_domain/lib/sipactivator.jar.>
<Jun 4, 2013 6:58:30 PM EDT> <Info> <WebLogicServer> <BEA-000377> <Starting WebLogic Server with OpenJDK 64-Bit Server VM Version 23.2-b09 from Oracle Corporation>
<Jun 4, 2013 6:58:31 PM EDT> <Info> <Management> <BEA-141107> <Version: WebLogic Server 10.3.6.0 Tue Nov 15 08:52:36 PST 2011 1441050 >
<Jun 4, 2013 6:58:32 PM EDT> <Info> <Security> <BEA-090065> <Getting boot identity from user.>
Enter username to boot WebLogic server: █
```

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

The following is used as an example:

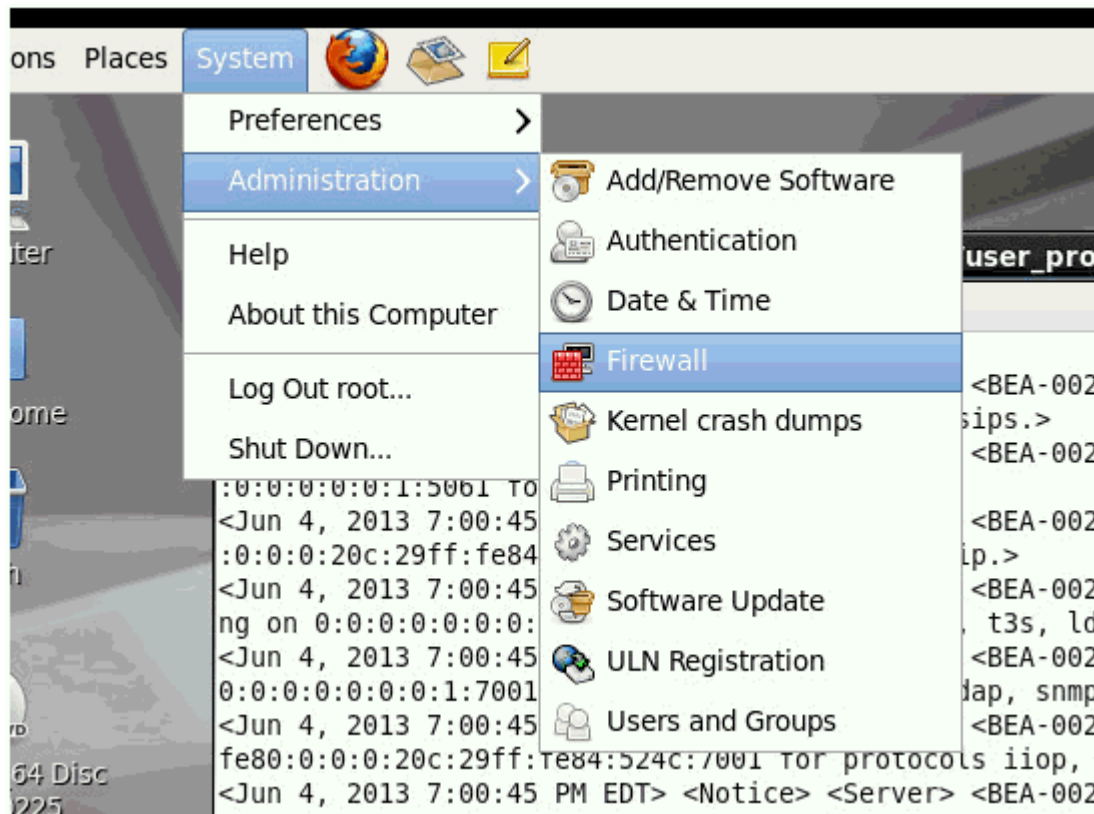
- Name:** weblogic
- User password:** WebLogic!! ("0" is a zero)

```
root@occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
File Edit View Search Terminal Help
0:0:0:0:1:5060 for protocols sip.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[2]" is now listening on fe80:0:0:0:20c:29ff:fe84:524c:5061 for protocols sips.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[4]" is now listening on 0:0:0:0:0:1:5061 for protocols sips.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sip[2]" is now listening on fe80:0:0:0:20c:29ff:fe84:524c:5060 for protocols sip.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure[4]" is now listening on 0:0:0:0:0:1:7002 for protocols iiops, t3s, ldaps, https.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "Default[4]" is now listening on 0:0:0:0:0:1:7001 for protocols iiop, t3, ldap, snmp, http.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "Default[2]" is now listening on fe80:0:0:0:20c:29ff:fe84:524c:7001 for protocols iiop, t3, ldap, snmp, http.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure[2]" is now listening on fe80:0:0:0:20c:29ff:fe84:524c:7002 for protocols iiops, t3s, ldaps, https.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[1]" is now listening on 192.168.122.1:5061 for protocols sips.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sip[1]" is now listening on 192.168.122.1:5060 for protocols sip.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure[1]" is now listening on 192.168.122.1:7002 for protocols iiops, t3s, ldaps, https.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <WebLogicServer> <BEA-000329> <Started WebLogic Admin Server "AdminServer" for domain "base domain" running in Production Mode>
Jun 04, 2013 7:00:45 PM oracle.sdp.common.cluster.util.ConfigReaderUtil getAdminSvrInstByCluster
INFO: The DCU is running in cluster server environment!
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport UDP)" is listening on port 5060>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
```

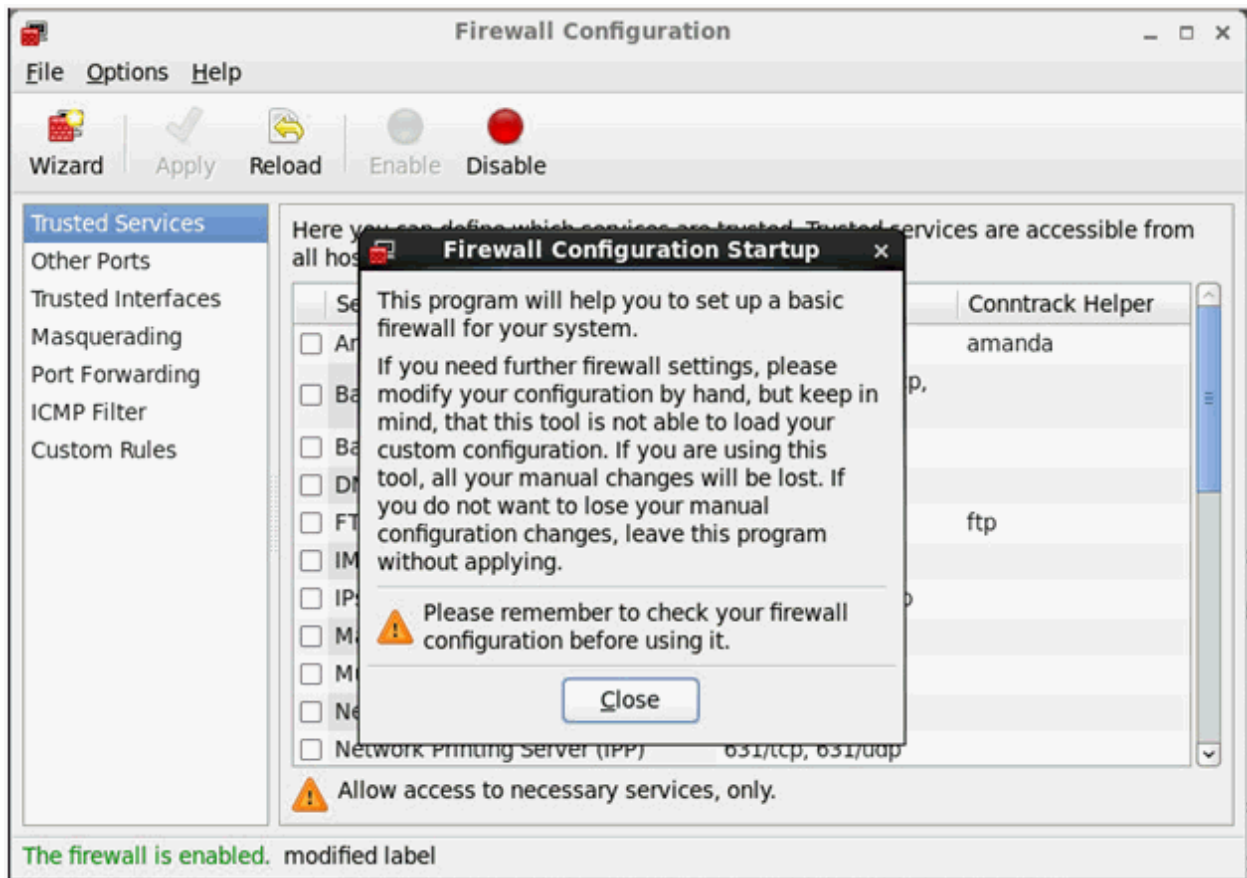
To verify that OCCAS is started, check if **<Server started in RUNNING mode>** is displayed.

### Firewall Configuration

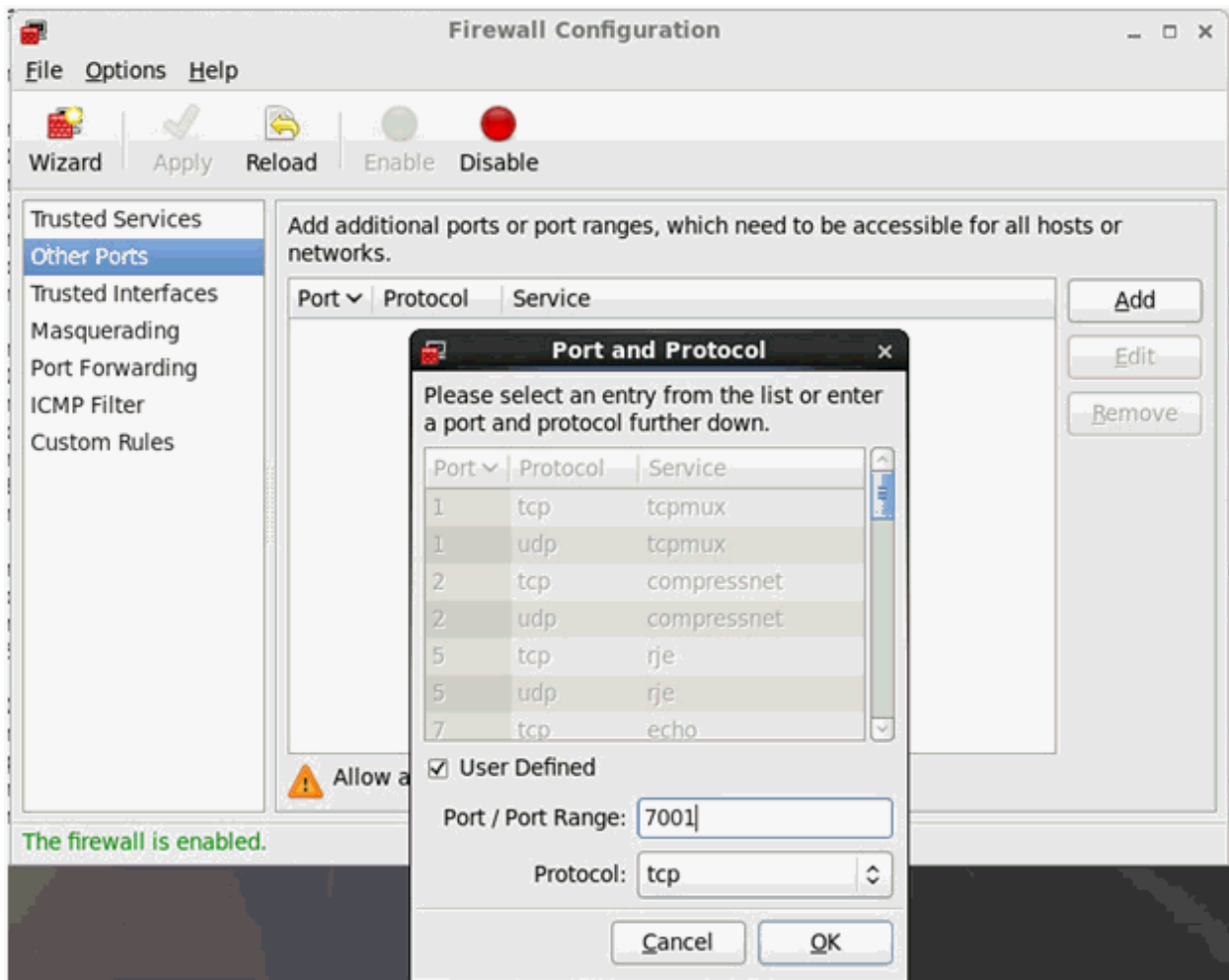
Enable port 7001/tcp, 5060/udp, and 5061/udp in the Linux firewall.



Go to **System > Administration** and select **Firewall**.

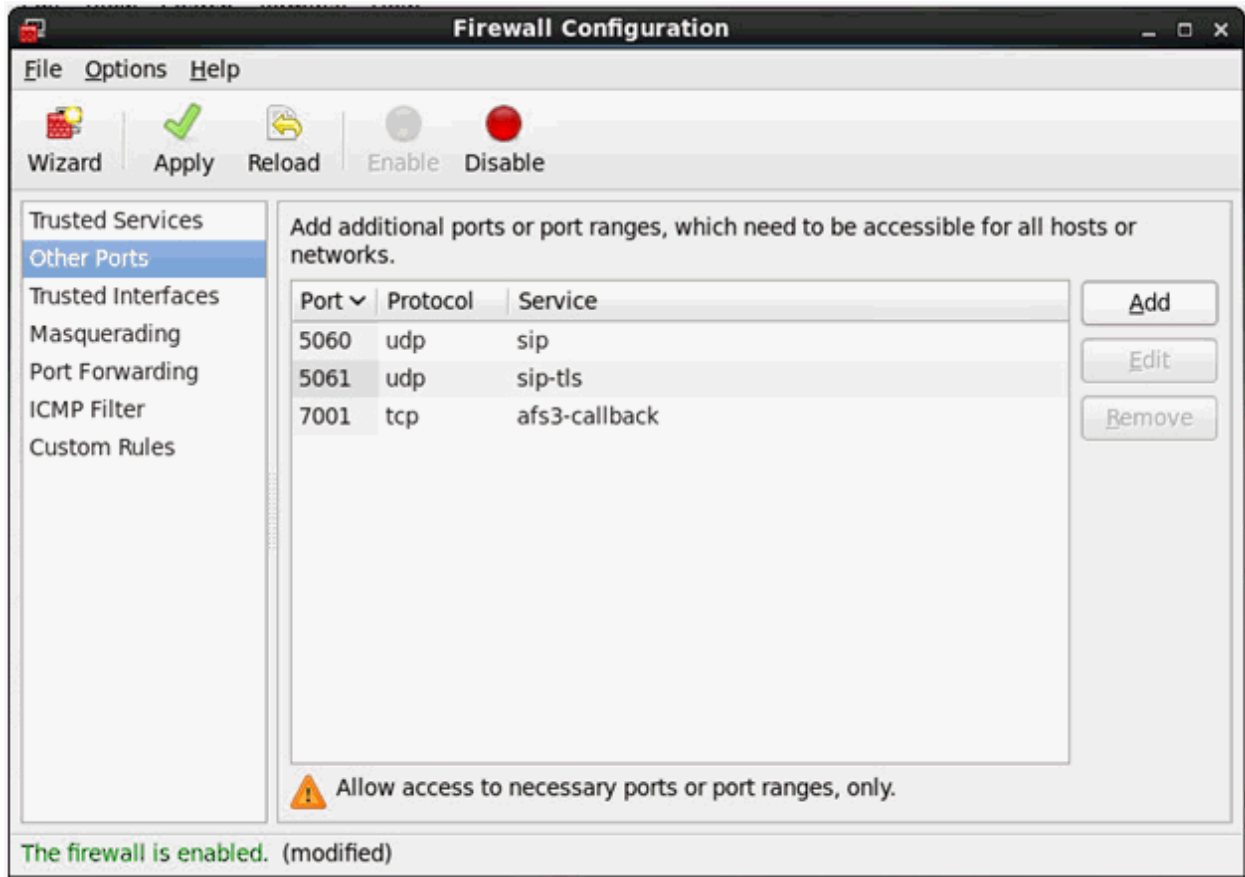


Click on **Close**.

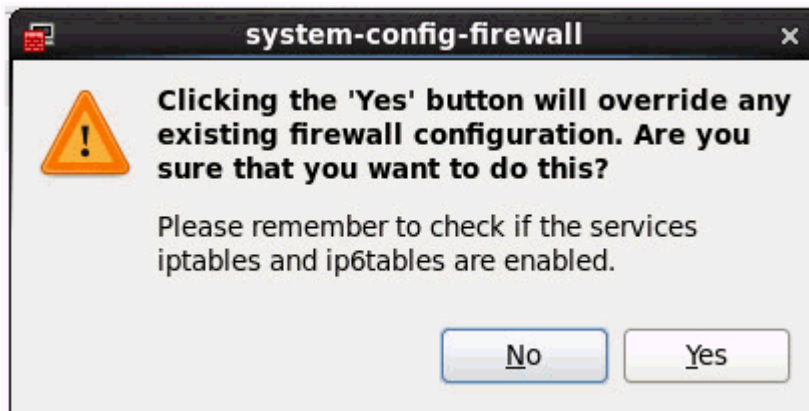


Select **Other Ports**, then click on **Add** button. Check the **User Defined** box, enter **7001** for **Port/Port Range**, choose **tcp** for **Protocol**, and then click **OK**.

Repeat the steps to add **5060** and **5061** for **Port/Port Range** but with **udp** as the **Protocol** for each.



Click on **Apply**.



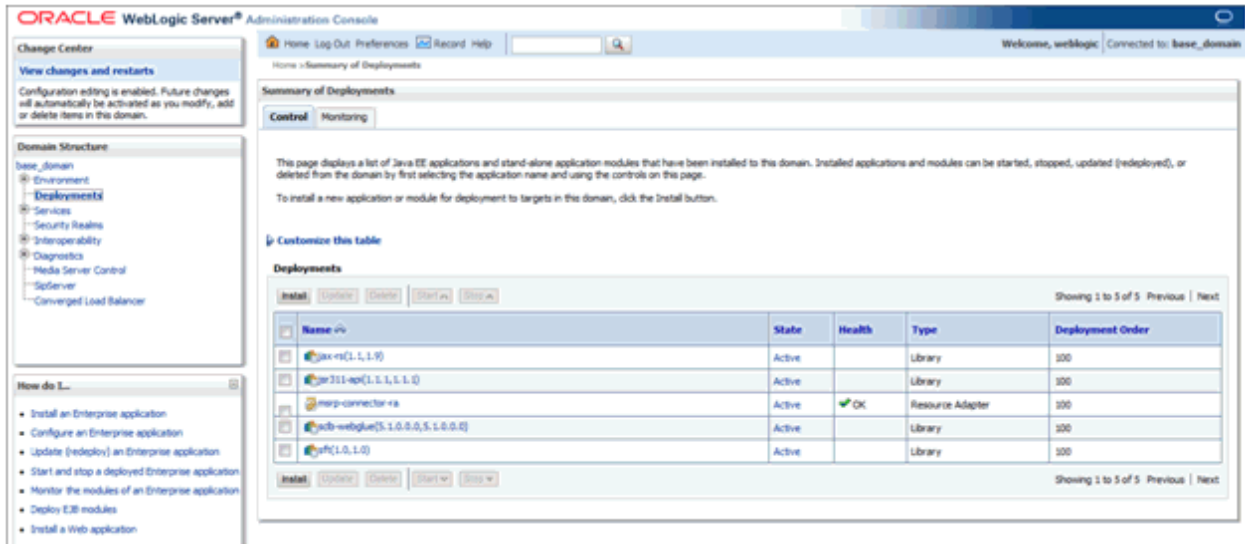
Click on **Yes** to activate the **Firewall Configuration**.

Now exit the **Firewall Configuration**.

## OCCAS Verification

Access the **Administration Console** to verify the installation at:

http://<as\_ip\_address>:7001/console



The screenshot shows the Oracle WebLogic Server Administration Console interface. The main content area is titled "Summary of Deployments" and contains a table of installed applications and modules. The table has columns for Name, State, Health, Type, and Deployment Order. The table lists five items, all with a State of "Active".

Name	State	Health	Type	Deployment Order
jax-ws(1.1.1.0)	Active		Library	100
jax-311-impl(1.1.1.1.0)	Active		Library	100
soap-converter-ra	Active	OK	Resource Adapter	100
wls-wsdl-wsdl(1.0.0.0,5.1.0.0.0)	Active		Library	100
ws(1.0.1.0)	Active		Library	100

Go to **Domain Structure** and click on **Deployments** to make sure **State** and **Health** are similar to the screen shot above.



## 8. Appendix B: Redundant Media Servers Configuration

---

The Redundant Media Server feature provided by the JSR 309 Connector supports hot active/standby redundancy. The JSR 309 Connector allows for "n" number of PowerMedia XMS systems to be configured where only one is an active Media Server and the rest are considered standby. This section explains how to configure hot active/standby.

A primary (hot active) or a single (redundancy is not used) PowerMedia XMS is defined in the *dlgc\_JSR309.properties* file:

```
##### Dialogic PowerMedia XMS Media Server Configuration #####
# Configuration of PowerMedia XMS Media Server
mediaserver.1.sip.address=xxxx.xxx.xxx.xxx
mediaserver.1.sip.port=xxx

# mediaserver.count defines the number of PowerMedia XMS Media Servers used
# by the JSR 309 Connector.
# Supported values:
# 1: Specifies single Media Server configuration (Redundancy not used)
# <2-n>: Specifies ALL Media Servers to be used by connector (Redundancy ON).
#     NOTE: Requires Redundancy Configuration section to be configured.
# Default - 1:
mediaserver.count=1
```

Redundancy configuration can be found under the "Dialogic PowerMedia XMS MS Redundancy Configuration" section in the *dlgc\_JSR309.properties* file:

```
##### Dialogic PowerMedia XMS MS Redundancy Configuration #####
# mediaserver.redundancy - turns redundancy feature "on" or "off"
# Default - off
mediaserver.redundancy=off

# Configuration of secondary set of PowerMedia XMS Media Server(s):
# NOTE: Configuration of primary PowerMedia XMS Media Server is defined in
# JSR 309 Connector Configuration section above as mediaserver.1.
# 1) Replicate the two lines below for each PowerMedia XMS used as secondary Media Server
# 2) change mediaserver.x to the next appropriate index
# 3) configure appropriate IP and PORT for each
# NOTE: number of Media Servers defined below has to match mediaserver.count parameter.
mediaserver.x.sip.address=xxx.xxx.xxx.xxx
mediaserver.x.sip.port=xxx

# mediaserver.redundancy.check.interval (in milliseconds) defines a time interval used by
# by JSR 309 Connector for sending a keep alive ping
# Default - 5000
mediaserver.redundancy.check.interval=5000

# mediaserver.redundancy.nonprimary.discover.clock.cycle defines a number of cycles to delay
# keep alive ping for every secondary Media Server(s)
# NOTE: cycle is used for secondary Media Servers and only used on initial discovery,
# i.e., startup of JSR 309 Connector.
# cycle * interval = seconds to wait before pinging secondary Media Server
# Default - 1
mediaserver.redundancy.nonprimary.discover.clock.cycle=1

##### END - Dialogic PowerMedia XMS MS Redundancy Configuration #####
```

In the “Dialogic PowerMedia XMS Media Server Configuration” section:

- mediaserver.1.sip.address and mediaserver.1.sip.port need to be configured for hot active Media Server.
- mediaserver.count needs to specify a total number of Media Servers to be used by a connector (one designated as active hot and others designated as active standbys).  
For example, if there are 4 Media Servers to be used where one of them is hot active and other 3 are considered hot standbys mediaserver.count needs to be set to 4 (1 hot active and 3 hot standbys).

In the “Dialogic PowerMedia XMS MS Redundancy Configuration” section:

- mediaserver.redundancy needs to be set to “on”.
- mediaserver.x.sip.address/port set of parameters need to be configured for hot standby Media Servers where x is from 2-4 (if total of 4 Media Servers are used).

Optional:

- mediaserver.redundancy.check.interval parameter defines a time interval in milliseconds for the JSR 309 Connector to send a keep alive ping to all configured Media Servers.

```
Default - 5000
```

- mediaserver.redundancy.nonprimary.discover.clock.cycle parameter defines a number of cycles to delay a keep alive ping for every hot standby Media Server.

**Note:** The cycle is used for secondary Media Servers and only used on initial discovery (i.e., startup of the JSR 309 Connector).

```
cycle * interval = seconds to wait before pinging hot standby Media Server  
# Default - 1
```

For details on how the JSR 309 Connector Media Server Redundancy works, refer to the Redundant Media Servers Guidelines section in the *Dialogic® PowerMedia™ JSR 309 Connector Software Developer’s Guide*.

## 9. Appendix C: Updating the JSR 309 Connector

The JSR 309 Connector comes as a set of JAVA library files (JAR). In the OCCAS 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 "DOMAIN\_HOME" directory.

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

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

- *dlgmsc.jar*
- *msmltypes.jar*
- *dlgcsmltypes.jar*

The *MANIFEST.MF* file has been included in the JSR 309 Connector JAR files with version and build number. The versions of these files need to be exactly the same and should not be mixed and matched with older or newer versions of the JAR files. In addition, the *MANIFEST.MF* file describes the version of PowerMedia XMS that the JSR 309 Connector was tested on.

In order for the new JAVA library JAR file to take effect, the application using it will have to be stopped and restarted through the OCCAS administration page (**Deployments** under **Domain Structure**) as shown below:

The screenshot shows the Oracle WebLogic Server Administration Console. The main content area is titled "Summary of Deployments" and contains a table of installed applications and modules. The table has the following data:

Name	State	Health	Type	Deployment Order
dlgmsc_tests	Prepared	OK	Web Application	100
jax-rs(1.1,1.9)	Active		Library	100
jsr311-api(1.1.1,1.1.1)	Active		Library	100
msrp-connector-ra	Active	OK	Resource Adapter	100
sclb-webglue(5.1.0.0,5.1.0.0)	Active		Library	100
sft(1.0,1.0)	Active		Library	100