



# **JSR 309 Connector Software for Dialogic® PowerMedia™ XMS**

**User's Guide**

# Copyright and Legal Notice

---

Copyright © 2010-2014 Dialogic Inc. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Inc. 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 Inc. 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 Inc. 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 Inc. 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. Welcome .....</b>	<b>7</b>
Assumptions .....	7
Related Information .....	7
<b>2. Overview .....</b>	<b>8</b>
Terminology .....	8
JSR 309 Media Server Control API .....	8
JSR 309 Connector .....	9
System Overview .....	9
<b>3. Supported Features .....</b>	<b>10</b>
Limitations .....	10
Supported Configurations .....	11
<b>4. System Requirements .....</b>	<b>12</b>
Hardware and Software Requirements .....	12
Supported Platforms .....	12
<b>5. Contents of the Distribution .....</b>	<b>13</b>
Distributed Files .....	13
Demo Prompts .....	13
<b>6. Installation and Configuration .....</b>	<b>14</b>
Preparing the J2EE Converged Application Server .....	14
Installing the JSR 309 Connector .....	14
<b>7. Test Servlets .....</b>	<b>17</b>
Test Servlets Overview .....	17
Configuring the Test Servlets .....	17
Installing the Demo Prompts .....	17
Running the Test Servlets .....	18
DlgcPlayerTest .....	18
DlgcDtmfPromptAndCollectTest .....	18
DlgcRecorderTest .....	19
DlgcDtmfAsyncTest .....	19
Conference Demos .....	20
JMCConferenceServlet .....	20
DlgcReferenceConferenceWithOutBCallServlet .....	21
DialogicBridgeConference .....	22
DlgcEarlyMediaBridgeDemo .....	22
<b>8. Developer Guidelines .....</b>	<b>25</b>
General Guidelines .....	25
Early Media Guidelines .....	25
Redundant Media Servers Guidelines .....	25
Configuring Network Time Protocol (NTP) for Accurate SIP Timers .....	26
Configuring the Application Server .....	26
Using Serialization .....	26
Disabling Serialization .....	26
Guidelines for Using Serialization .....	27
Configuration for a Distributed Environment .....	28
Example Code .....	28
Example A .....	28
Example B .....	29

<b>9. Troubleshooting .....</b>	<b>30</b>
Logging.....	30
SIP Errors.....	30
<b>10. Appendix: JSR 309 Connector Environment Setup .....</b>	<b>31</b>
Installing and Configuring the OCCAS .....	31
Pre-Installation Setup .....	32
JDK Setup .....	32
OCCAS Installation .....	33
OCCAS Configuration .....	44
OCCAS Startup.....	54
Firewall Configuration.....	57
OCCAS Verification .....	61
Installing the JSR 309 Connector.....	62
Installing and Configuring the Test Servlets .....	64
Running the Test Servlets.....	74

## Revision History

---

Revision	Release Date	Notes
05-2708-006	February 2014	Updates to support PowerMedia XMS Release 2.2.
05-2708-005	October 2013	<p>Updates to support PowerMedia XMS Release 2.1.</p> <p>Global change:</p> <ul style="list-style-type: none"><li>Renamed this document from Quick Start Guide to User's Guide.</li></ul> <p><a href="#">System Requirements:</a></p> <ul style="list-style-type: none"><li>Updated to include support for OCCAS 5.1.0.</li></ul> <p><a href="#">Test Servlets:</a></p> <ul style="list-style-type: none"><li>Added, removed, and updated demos.</li><li>Added new section for Configuring the Test Servlets.</li><li>Updated instructions for demo prompts.</li></ul> <p><a href="#">Appendix: JSR 309 Connector Environment Setup:</a></p> <ul style="list-style-type: none"><li>Updated to include support for OCCAS 5.1.0.</li></ul>
05-2708-004	April 2013	<p><a href="#">Supported Features:</a></p> <ul style="list-style-type: none"><li>Added new supported configurations in <a href="#">Supported Configurations</a> table.</li></ul> <p><a href="#">Test Servlets:</a></p> <ul style="list-style-type: none"><li>Added new section for DlgcReferenceConferenceDemo.</li></ul>
05-2708-003	March 2013	<p><a href="#">Supported Features:</a></p> <ul style="list-style-type: none"><li>Added new video file playback and video conference features.</li></ul>
05-2708-002	February 2013	<p><a href="#">Contents of the Distribution</a> and <a href="#">Installation and Configuration:</a></p> <ul style="list-style-type: none"><li>Added new files to support Simple Logging Facade framework.</li></ul>

Revision	Release Date	Notes
05-2708-001	January 2013	<p>Updates to support PowerMedia XMS Release 2.0.</p> <p>Global change:</p> <ul style="list-style-type: none"> <li>Renamed Dialogic® PowerMedia™ Media Server Connector (PowerMedia MSC) to JSR 309 Connector Software for Dialogic® PowerMedia™ XMS (JSR 309 Connector).</li> <li>Integrated content from Release Notes into this Quick Start Guide.</li> </ul> <p><b>Supported Features:</b></p> <ul style="list-style-type: none"> <li>Added new bridge conference feature.</li> </ul> <p><b>System Requirements:</b></p> <ul style="list-style-type: none"> <li>Added new section for hardware and software requirements along with supported platforms.</li> </ul> <p><b>Installation and Configuration:</b></p> <ul style="list-style-type: none"> <li>Added new content for installation and configuration.</li> </ul> <p><b>Test Servlets:</b></p> <ul style="list-style-type: none"> <li>Added new section for DlgcEarlyMediaBridgeDemo.</li> </ul> <p><b>Appendix: JSR 309 Connector Environment Setup:</b></p> <ul style="list-style-type: none"> <li>Added new section for instructions on how to set up the JSR 309 Connector environment.</li> </ul>
05-2708-001-01	October 2012	Updates to support PowerMedia MSC Release 3.0 Beta.
Version 0.3	April 2011	Updates to support PowerMedia MSC Release 1.0 Beta - Updates.
Version 0.2	December 2010	Updates to support PowerMedia MSC Release 1.0 Beta.
Version 0.1	October 2010	Initial release of this document.
Last modified: February 2014		

Refer to [www.dialogic.com](http://www.dialogic.com) for product updates and for information about support policies, warranty information, and service offerings.

# 1. Welcome

---

This User's Guide is written for users of the JSR 309 Connector Software for Dialogic® PowerMedia™ XMS (also referred to herein as "JSR 309 Connector"), which is used in conjunction with the Dialogic® PowerMedia™ Extended Media Server (also referred to herein as "PowerMedia XMS" or "XMS").

This User's Guide describes the JSR 309 Connector, provides installation and configuration information, and describes the test servlets included in this release.

## Assumptions

This User's Guide assumes that you have the following knowledge and experience:

- Familiarity with the Java Specification Request (JSR) 309 documentation version 1.0.
- Familiarity with Oracle Communications Converged Application Server (OCCAS) development and administration.
- Prior experience with JSR 289 (SIP Servlets).
- Prior experience with Java Platform Enterprise Edition (Java EE) development.
- Familiarity with PowerMedia XMS administration and configuration.

## Related Information

See the following for more information:

- PowerMedia XMS datasheet at [www.dialogic.com](http://www.dialogic.com).
- PowerMedia XMS documentation at [www.dialogic.com/manuals](http://www.dialogic.com/manuals).
- Dialogic technical support at [www.dialogic.com/support](http://www.dialogic.com/support).
- JSR 309 documentation on the Java Community Process website at [www.jcp.org](http://www.jcp.org).

## 2. Overview

---

This section provides an overview of the Java Specification Request (JSR) 309 and describes the JSR 309 Connector features and limitations.

### Terminology

A brief description of terminology used in this document is provided for reference.

**Servlet** – A Java class which conforms to the Java Servlet Interface, by which a Java class may respond to HTTP requests. Applications using servlets may be packaged in a WAR file as a web application.

**Java Specification Request (JSR)** – A formal document created by members of the Java Community Process (JCP) that adds features and functionality to the Java platform.

**JEE or J2EE** – Java Platform, Enterprise Edition. A platform for server programming in the Java programming language.

**Web Application Server** – Application Server based on JEE or J2EE.

**MSML** – Media Server Markup Language.

**OCCAS** – Oracle Communications Converged Application Server.

**Conference Control Leg** – This terminology describes when a conference is created using a mixer component and the mixer is configured to create a conference control leg. A conference control leg requires the use of one connection resource from the Media Server.

**No Conference Control Leg** – This terminology describes when a conference is created using a mixer component and the mixer is configured to **not** create a conference control leg. In this case, the JSR 309 Connector uses the call legs to send conference configuration. Therefore, it does not require an extra Media Server resource for conference control.

### JSR 309 Media Server Control API

Java Specification Request (JSR) 309 is a standard Java media server control API for multimedia application development. It provides a generic media server abstraction interface that is independent of the underlying media server control protocol. The multimedia applications, such as IVR, voice-mail, audio conferencing, and call center, are typically deployed in a SIP-based infrastructure.

The JSR 309 API provides three areas of functionality:

- Network Connection to establish media streams.
- Media Group functions such as to play, record, and control media content.
- Media Mixer functions to join media functions to a network connection so as to create conferences and call bridges.

For details on the JSR 309 API, see the documentation on the Java Community Process website at [www.jcp.org](http://www.jcp.org).

For a list of JSR 309 API parameters supported by the JSR 309 Connector, see [Supported Features](#).

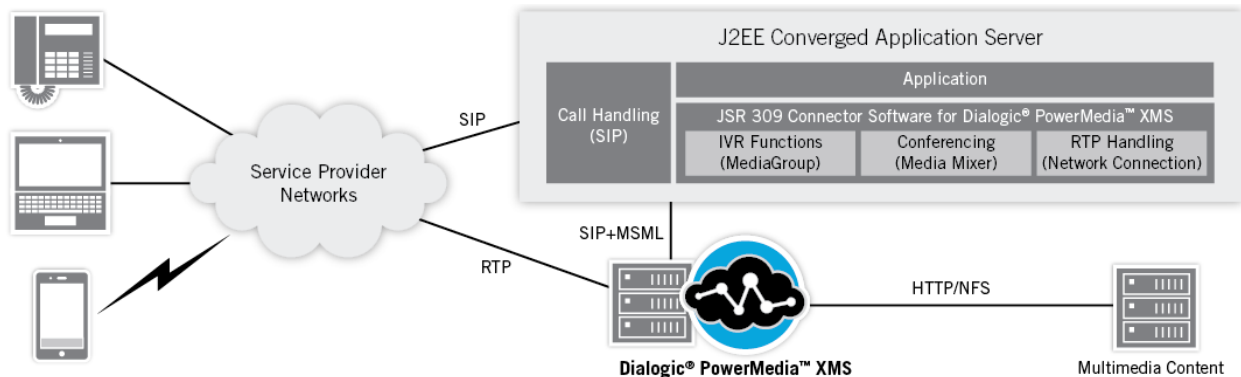
## JSR 309 Connector

The JSR 309 Connector is the Dialogic implementation of the JSR 309 version 1.0 final specification. This software runs on application servers such as the OCCAS and enables a multimedia application on the application server to control the PowerMedia XMS using the JSR 309 API.

The JSR 309 Connector is designed to support an asynchronous programming model. Applications using the connector should be designed to interface using Listener objects for events on operation completion.

## System Overview

The following figure illustrates the role of the JSR 309 Connector in a typical deployment.



The following components are included in this figure:

**J2EE Converged Application Server** – Handles SIP call control and other aspects of real-time multimedia communications using a Java EE environment with JSR 289 support.

**Application** – Runs on the J2EE Converged Application Server. Examples of applications: IVR, conferencing, announcements, and call centers.

**JSR 309 Connector** – Software connector that enables the J2EE Converged Application Server to control PowerMedia XMS through JSR 309-compliant API calls.

**PowerMedia XMS** – Performs the multimedia operations required to establish and maintain real-time communications while providing a high-quality user experience.

**External Servers** – Used for storing and streaming multimedia content.

**SIP Servlet API for Call Handling** – SIP stack used to communicate with SIP compliant user agents. JSR 289 is the standard API servlet used by Converged Communication Application Servers.

## 3. Supported Features

---

The JSR 309 Connector is compatible with the JSR 309 Media Server Control API version 1.0.

The JSR 309 Connector supports the following functionality:

- Driver loading with driver property support
- Audio file playback
- Video file playback
- Audio recording
- Conference mixing (n-party mixing based on loudest talker)
- Bridge conference
- Basic prompt and collect
- Video conference
- Signal detection
- Redundant media servers (active and alternate)
- Serialization
- Cluster support
- Video streaming to network connections
- Media mixing with video layout
- Media play/record
- Media handling for WebRTC

Video streams, video layout, and video rendering components are not supported.

### Limitations

The following high-level functionality is **not** implemented in the JSR 309 Connector:

#### Conference Limitations

Side-bar and coach-pupil conference to conference formats are not supported.

#### Runtime Controls

Various JSR 309 APIs accept Runtime Controls (RTC) as formal parameters. Overall, RTC is not supported, except for the following which are translated to barge-in functionality:

- RTC(SignalDetector.DETECTION\_OF\_ONE\_SIGNAL, Recorder.STOP)
- RTC(MediaGroup.SIGDET\_STOPPLAY, <n/a>)

#### Signal Generator

Signal Generator API is not supported.

#### CodecPolicy

The CodecPolicy implementation is not supported. Thus, no codec filtering is done at the connector level.

## VxmlDialog

VxmlDialog is not supported.

## Supported Configurations

The JSR 309 API has three core JSR API resource containers, NC, MG, and MX, that can be joined together in three different modes, SEND, RECV, and DUPLEX. However, not all combinations are supported.

NC refers to Network Connection and provides functionality to establish media streams. MG refers to Media Group and provides functionality to play, record, and control media content. MX refers to Media Mixer and provides functionality to join media functions to a network connection so as to create conferences and call bridges.

SEND means the media streams can flow from joiner to joinee only. RECV means the media streams can flow from joinee to joiner only. DUPLEX means the media streams can flow both ways.

The following table shows the supported configurations:

	<b>NC</b>	<b>MG</b>	<b>MX</b>
<b>NC (DUPLEX)</b>	YES	YES	YES
<b>NC (SEND)</b>	YES	NO	YES
<b>NC (RECV)</b>	YES	NO	YES
<b>MG (DUPLEX)</b>	YES	NO	YES
<b>MG (SEND)</b>	NO	NO	NO
<b>MG (RECV)</b>	NO	NO	NO
<b>MX (DUPLEX)</b>	YES	YES	NO
<b>MX (SEND)</b>	YES	NO	NO
<b>MX (RECV)</b>	YES	NO	NO

The following join combinations are the same and can be used to set up a full duplex connection. In the examples below, a Media Mixer is connected in full duplex to a Network Connection for play/record capability.

```
networkConnection.join(Direction.RECV, mixer)
mixer.join(Direction.SEND, networkConnection)
```

```
networkConnection.join(Direction.SEND, mixer)
mixer.join(Direction.RECV, networkConnection)
```

```
networkConnection.join(Direction.DUPLEX, mixer)
mixer.join(Direction.DUPLEX, networkConnection)
```

As per the JSR 309 Connector implementation for any above join combination, please make note that the Allocation Event is always sent to the Network Connection component.

## 4. System Requirements

---

### Hardware and Software Requirements

This User's Guide assumes that you have the following systems in place before installing JSR 309 Connector:

- A working OCCAS 5.1.0 system for development and testing.
- A working PowerMedia XMS Release 2.2 system.
- SIP phones or soft clients.

### Supported Platforms

The JSR 309 Connector was developed using the Java SDK version 1.6.0. The JSR 309 Connector has been deployed and tested on OCCAS 5.1.0.

## 5. 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 the *JSR309\_MSC3\_BuildXX.tar* file. This package contains the following directories:

- *DlgcJSR309* - consists of various required components for the JSR 309 Connector as well as demos.
- *DlgcJSR309DemoPrompts* - consists of media files included in the JSR 309 Connector distribution.

**Note:** <Release Package> refers to the *.tar* distribution package provided for this release.

JSR 309 Connector Files	Description
<Release Package>/DlgcJSR309/applications/ <i>dlgmsc_tests.war</i>	Directory that contains a deployable web archive that can be used to test the supported functionality. The WAR file implements several test servlets; see <a href="#">Test Servlets</a> for more information.
<Release Package>/DlgcJSR309/lib/ <i>dlgmsc.jar</i> <i>jain-sip-sdp-1.2.91.jar</i> <i>jsr173_1.0_api.jar</i> <i>log4j-1.2.15.jar</i> <i>msmltypes.jar</i> <i>slf4j-api-1.7.2.jar</i> <i>slf4j-log4j12-1.7.2.jar</i> <i>xbean.jar</i>	Directory that contains the JSR 309 Connector implementation for PowerMedia XMS and required third-party software libraries.
<Release Package>/DlgcJSR309/properties/ <i>dlgcsr309.properties</i> <i>dlgcsr_demos.properties</i> <i>log4j.properties</i>	Directory that contains the properties files used to set up configuration for JSR 309 Connector, the provided demos, and common logging facility.
<Release Package>/DlgcJSR309/sample-src/ <i>dlgmsc_tests.zip</i>	Directory that contains the test servlets source files. Test servlets are only provided for reference. Please read the copyright notice included with the test servlets.

### Demo Prompts

The demo prompts, to run test servlets, consists of media files included in the JSR 309 Connector distribution. To install the demo prompts, see [Installing the Demo Prompts](#).

## 6. Installation and Configuration

---

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

For system requirements and supported platforms, see [System 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.

### Installing the JSR 309 Connector

Follow this procedure to get the application (WAR file) in an OCCAS environment to correctly load the JSR 309 Connector (*dlgmsc.jar*).

The JSR 309 Connector supports PowerMedia XMS via the MSML protocol.

#### Step 1

Extract the *JSR309\_MSC3\_BuildXX.tar* distribution package for this release which will create a <Release Package> directory.

**Note:** <Release Package> refers to the .tar distribution package provided for this release.

Copy all the .jar files from <Release Package>/DlgsJSR309/lib directory to the OCCAS <Domain Location>/lib directory.

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

JAR Files	Description
<i>dlgmsc.jar</i> <i>jain-sip-sdp-1.2.91.jar</i> <i>jsr173_1.0_api.jar</i> <i>log4j-1.2.15.jar</i> <i>msmltypes.jar</i> <i>slf4j-api-1.7.2.jar</i> <i>slf4j-log4j12-1.7.2.jar</i> <i>xbean.jar</i>	JSR 309 Connector implementation for PowerMedia XMS and required third-party software libraries.

## Step 2

Set up the properties files.

JSR 309 Connector uses the following properties files:

- `dlgcJSR309.properties` - used to configure IP addresses and ports for the OCCAS environment using JSR 309 Connector and dedicated PowerMedia XMS platform.
- `log4j.properties` - used for logging using Simple Logging Facade framework implementation of log4j.

**Note:** You can configure logging in the `log4j.properties` file. By default, logging is configured to use INFO level and output to the Console as well as to a `dlgmsc.log` file.

Follow these steps to set up the properties file:

1. Copy the `dlgcJSR309.properties` and `log4j.properties` files from <Release Package>/DlgcJSR309/properties directory to the OCCAS <Domain Location>/config directory.
2. Edit the `dlgcJSR309.properties` file according to your OCCAS and PowerMedia XMS configuration. The changes will include the OCCAS IP address and SipServlet container port running the JSR 309 Connector as well as the PowerMedia XMS IP address and port:  
 # JSR 309 Connector address information (typically same as the SipServlet container)  

```
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5060
```

  
 # PowerMedia XMS address information  

```
mediaserver.1.sip.address=xxx.xxx.xxx.xxx
mediaserver.1.sip.port=5060
```
3. Edit the <Domain Location>/bin/startWebLogic.sh OCCAS startup script. Look for the following line:  

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

Add the following lines directly after:

```
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/dlgcJSR309.properties
LOG4J_OPTIONS="-Dlog4j.configuration=${DOMAIN_HOME}/config/log4j.properties"
XQUERYPATH=/root/Oracle/Middleware/modules/features/weblogic.server.modules.xquery_10.3.6
.0.jar
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/server/lib/jsr309-descriptor-binding.jar:${XQUERYPATH}"
```

4. Add the following line in OCCAS startup script to enable some of the relevant items and to disable serialization (highlighted in bold):

```
${LOG4J_OPTIONS} -Dlog4j.debug -Dwlss.local.serialization=false
```

from:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
    echo "Starting WLS with line:"
    echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME}
-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} -
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} -
Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS}
${PROXY_SETTINGS} ${SERVER_CLASS} >"${WLS_REDIRECT_LOG}" 2>&1
fi
```

to:

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

For more details on disabling serialization, refer to [Disabling Serialization](#).

5. Save the startup script file then restart the WebLogic Server.

## 7. 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 are provided to illustrate the use of the JSR 309 Connector. These test servlets are included in the *dlgmsc\_tests.war*.

Please read the copyright notice included with the test servlets.

### Configuring the Test Servlets

1. Copy the *dlgmsc\_tests.war* file from <Release Package>/DlgcJSR309/applications directory to the OCCAS <Domain Location>/applications directory.
2. Copy the *dlgc\_demos.properties* file from <Release Package>/DlgcJSR309/properties directory to the OCCAS <Domain Location>/applications directory.
3. Edit the *startWebLogic.sh* file from <Domain Location>/bin directory and add the following line to point to demo properties file:

```
export DIALOGIC_DEMO_PROPERTY_FILE=${DOMAIN_HOME}/application/dlgc_demos.properties
```

See the following example with the line added to point to demo properties file (highlighted in bold):

```
export DIALOGIC_DEMO_PROPERTY_FILE=${DOMAIN_HOME}/application/dlgc_demos.properties
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/dlgcJSR309.properties
LOG4J_OPTIONS="-Dlog4j.configuration=${DOMAIN_HOME}/config/log4j.properties"
XQUERYPATH=/root/Oracle/Middleware/modules/features/weblogic.server.modules.xquery 10.3.6.0
.jar
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/server/
lib/jsr309-descriptor-binding.jar:${XQUERYPATH}"
```

### Installing the Demo Prompts

Sample test servlets use custom demo prompts which need to be installed on dedicated PowerMedia XMS platform for this configuration. You can locate and install the demo prompts by performing the following:

1. Copy all the *.tar* files inside the <Release Package>/DlgcJSR309DemoPrompts directory to the PowerMedia XMS machine under /var/lib/xms/media/en\_US/verification directory.
2. Untar each of the media files using the command; `tar -xvf <Filename>.tar`.

As an alternative method, you can install the demo prompts using the PowerMedia XMS web interface which allows you to upload one file at a time.

**Note:** When installing via the PowerMedia XMS web interface upload page, the directory structure must be identical to the unzip directory.

## Running the Test Servlets

Using a SIP phone, special SIP URIs are used to initiate the test servlets. This information is available in the *sip.xml* file, which is included in the *dlgmsc\_tests.war*. Verify that the media files have been installed on PowerMedia XMS.

You can locate and access the *sip.xml* file by performing the following:

1. Create a temp directory.
2. Copy the *dlgmsc\_tests.war* and then run `jar -xvf dlgmsc_test.war` inside the temp directory.
3. Access the file contents located in `./WEB-INF/sip.xml`.

### 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@<OCCAS5.1.0-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@<OCCAS5.1.0-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@<OCCAS5.1.0-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@<OCCAS5.1.0-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 managed multiple conferences.	Media files needs to be installed in the PowerMedia XMS for menu to work.
DlgcAvConferenceDemo	Implements an advanced conference.	Media files needs 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 needs to be installed in the PowerMedia XMS for menu to work.

### JMCConferenceServlet

This test servlet illustrates how to create and managed 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 *dlgmisc\_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 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@<OCCAS5.1.0-IP-ADDRESS>
```

## DlgcReferenceConferenceWithOutBCallServlet

This test servlet illustrates how to implement an advanced conference that does not required a mixer control leg and 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 **DlgcAvConferenceDemo**. 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.

Menu Supported by DlgcAvConferenceDemo:

- \*00 - Plays announcement of menu options then goes back into conference
- \*01 - Allows the user to toggle between mute/unmute
- \*02 - Allows the user to unjoin the conference
- \*03 - Allows the user to rejoin the conference
- \*05 - Plays a song – Once the song completes, the leg goes back into conference
- \*06 - Plays announcement of "The conference size is"
- \*99 - Allows the user to stop any plays and return back to conference
- \*77 – Records the conference
- \*88 – Stops the conference recording
- \*44 – Makes an outbound call and joins the outbound call to the conference (requires configuration in the demo property file)

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 1:** 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:

```
DlgcAvConferenceDemo@<OCCAS5.1.0-IP-ADDRESS>
```

## DialogicBridgeConference

This test servlet illustrates how to implement a simple conference that does not require a mixer. That is two legs are directly joined into a conference.

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:

- It basically 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:

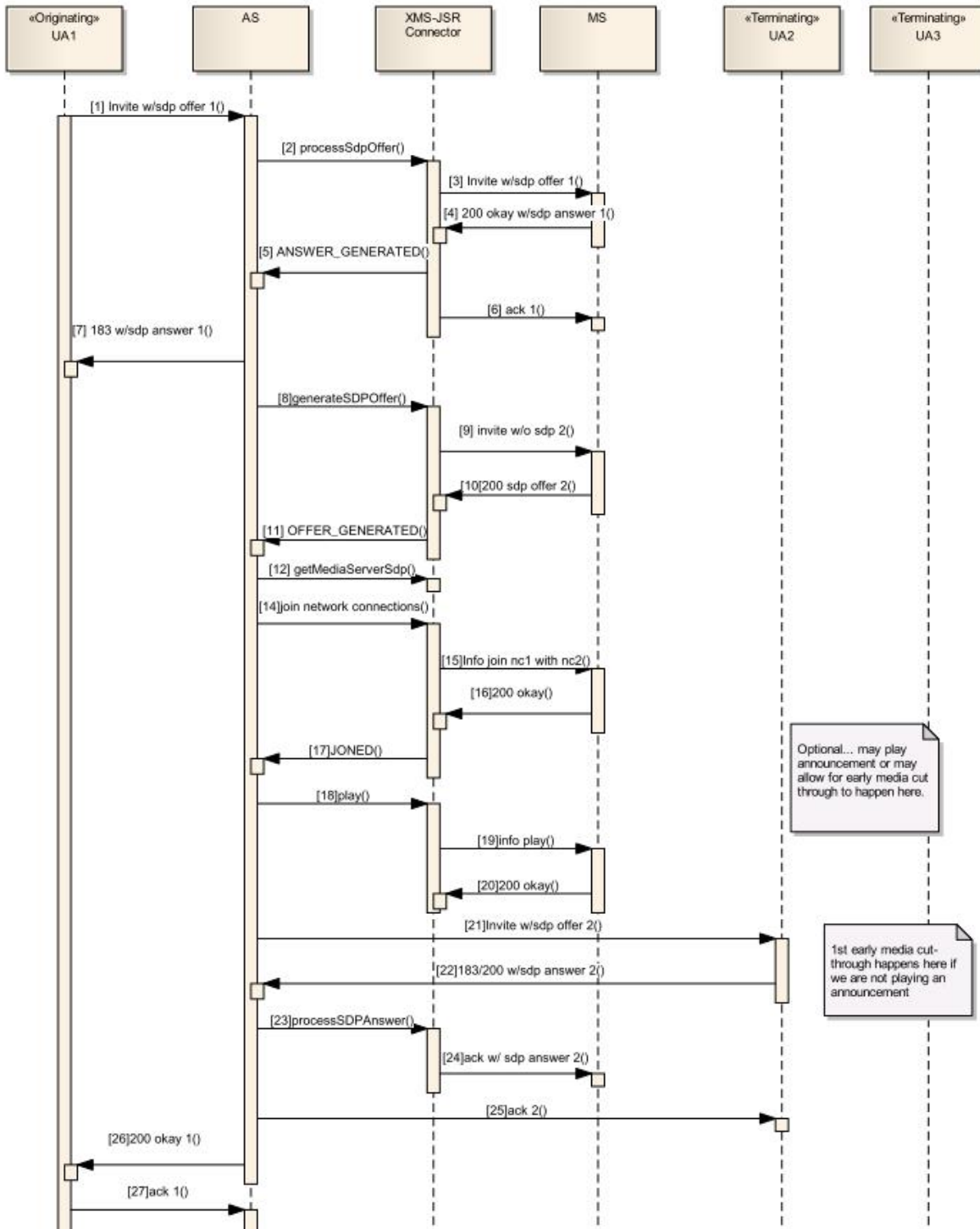
```
DlgcBridgeDemo@<OCCAS5.1.0-IP-ADDRESS>
```

## DlgcEarlyMediaBridgeDemo

This test servlet is similar to the DialogicBridgeConference defined above, except that it simulates an early media scenario. For more details on using early media, refer to [Early Media Guidelines](#).

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@<OCCAS5.1.0-IP-ADDRESS>
```

## 8. Developer Guidelines

---

This section provides application development guidelines for the JSR 309 Connector.

### General Guidelines

Consider the following guidelines when designing and implementing your application:

- Note that the `UNSOLICITED_OFFER_GENERATED` event type handling is no longer needed when moving a leg from conference to IVR with the JSR 309 Connector.
  - The application must handle the `UNSOLICITED_OFFER_GENERATED` event type. This event type is provided in the `SdpPortManagerEvent` object indicating that the media server has requested a change in the SDP; therefore, the application must send re-INVITE to its user agent.
- If your application does not release connector resources, you will likely lose resources on your PowerMedia XMS.

### Early Media Guidelines

Early media is supported by the JSR 289 and is enabled in the application.

To use early media, a media stream is established by the application while the SIP call leg is in 1xx state. Therefore, media will start flowing to the SIP endpoint before the call goes to the 2xx (connected) state.

A high-level early media call flow is as follows:

- From the JSR 289 (SIP side), the application waits for the INVITE with SDP offer.
- The application creates a Player using JSR 309 and passes in the SDP.
- The application sends the JSR 309 SDP answer back to the caller using a 183 response and using PRACK.
- The application instructs the PowerMedia XMS to play an announcement when the ACK (for the PRACK) is received.
- When the PowerMedia XMS has finished the announcement, it sends a BYE to the connector which signals the application.
- The application sends a 487 Call Terminated response to the caller rather than a 200 OK.

### Redundant Media Servers Guidelines

The JSR 309 Connector supports redundant (active/alternate) media servers.

If the primary (active) media server becomes non-responsive, the connector software automatically routes a new invite request to the next available redundant (alternate) media server. If no media servers are available, all new calls will fail until one of the configured media servers becomes available.

If the media server fails, the application will be notified via the `NETWORK_STREAM` error.

You may configure one or more media servers. The first media server is considered the primary (active) one and all others are used as redundant (alternate) media servers. The `mediaserver.redundancy` parameter in the properties file controls this feature. For more information, see [Installation and Configuration](#).

## Configuring Network Time Protocol (NTP) for Accurate SIP Timers

In order for the SIP protocol stack to function properly, you must accurately synchronize system clocks on all engine and SIP data tier servers to a common time source, to within one or two milliseconds. Differences in system clock settings can cause a number of severe issues such as:

- SIP timers firing prematurely on servers with the fast clock settings.
- Poor distribution of timer processing in the engine tier. For example, one engine tier server may process all expired timers, whereas other engine tier servers process no timers.

You should use a Network Time Protocol (NTP) client or daemon on each OCCAS instance and synchronize to a common NTP server.

Because the initial T1 timer value of 500 milliseconds controls the retransmission interval for INVITE request and responses, and also sets the initial values of other timers, even small differences in system clock settings can cause improper SIP protocol behavior. For example, an engine tier server with a system clock 250 milliseconds faster than other servers will process more expired timers than other engine tier servers, will cause retransmits to begin in half the allotted time, and may force messages to timeout prematurely.

## Configuring the Application Server

### Using Serialization

Java object serialization is used to support replication of Java objects to another process. Serialization enables an application in a distributed or clustered environment to support application replication. By default serialization is turned on in the JSR 309 Connector.

To use serialization in your application, follow these instructions:

1. Add the **transient** keyword to the JSR 309 MsControlFactory, Driver, and Listener instance attributes. See [Example A](#).

Note that the JSR 309 specification defines MsControlFactory, Driver, and Listener as non-serializable; therefore, the application is required to use the transient keyword. The application servlet that contains these attributes must implement the Java serializable interface.

2. All JSR 309 Listener classes must implement the serializable interface. See [Example B](#).

See additional guidelines in [Guidelines for Using Serialization](#).

### Disabling Serialization

Turning off serialization can improve performance. If your application does not need to participate in replication, you can disable local serialization by the container. To do so, add the following parameter to the OCCAS startup script:

```
-Dwlss.local.serialization=false
```

Following is the relevant portion of a startup script that has been modified to disable local serialization by the container.

```
if "%WLS_REDIRECT_LOG%"==" " (
    echo Starting WLS with line:

    echo %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%
-Dwlss.local.serialization=false -Dweblogic.Name=%SERVER_NAME%
-Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy %PROXY_SETTINGS% %SERVER_CLASS%

    %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dwlss.local.serialization=false
-Dweblogic.Name=%SERVER_NAME% -Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS%
) else (
    echo Redirecting output from WLS window to %WLS_REDIRECT_LOG%

    %JAVA_HOME%\bin\java %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS% -Dwlss.local.serialization=false
-Dweblogic.Name=%SERVER_NAME% -Djava.security.policy=%WL_HOME%\server\lib\weblogic.policy
%PROXY_SETTINGS% %SERVER_CLASS% >"%WLS_REDIRECT_LOG%" 2>&1
)
```

## Guidelines for Using Serialization

Follow these rules and guidelines for using serialization in your user application:

1. Avoid using instance and static variables in read and write mode because different instances may exist on different JVMs. In other words, static and transient attributes are not replicated.
2. All non-transient fields must be serializable (or primitive).
3. Always explicitly define the following in your serialized classes. For example:

```
private static final long serialVersionUID = <long integer>
```

You can use eclipse to generate a unique UID.

4. All classes in a hierarchy must be serializable. If the subclass implements serialization and the superclass does not, the subclass must take care of the serialization of the superclass attributes.
5. readObject implementations always start by calling the defaultReadObject() method.
6. The de-serialization process typically calls the readObject method. If you overwrite the readObject method, you must first call defaultReadObject() before adding any other instantiation to your object.
7. There is a performance cost to storing large object graphs in a Sip session or HTTP session. Large applications require using persistent sessions. Sessions must be read by the servlet whenever the object graph is used and rewritten whenever the object graph is updated.
8. Avoid using java.io.\* because the files may not exist on all application servers. Instead use getResourceAsStream.
9. Avoid storing values in a ServletContext. A ServletContext is not serializable and also the different instances may exist in different JVMs.

10. Use SipSession setAttribute method to store the session state and use SipSession getAttribute method to restore it. Note that the setAttribute and getAttribute methods respectively lock/unlock the call state. Remember that setAttribute and getAttribute are expensive operations so use these methods judiciously.
11. In OCCAS the SAS.doAction() method also locks/unlocks the call state.
12. When using Maps to store serializable information, both keys and data must be serializable. Note the following rule:
  - Do not use an object reference as a key, because the object hashed ID changes between the serialization and de-serialization operations. This means that searching the Map may fail.

## Configuration for a Distributed Environment

If your application needs to run in a distributed environment, you must add the <distributed/> tag in your application's *sip.xml* and make sure that serialization is set to true (that is, it is not set to false) in the OCCAS startup script.

## Example Code

### Example A

This example illustrates serialization best practices. The text in bold illustrates various rules and guidelines discussed in [Using Serialization](#) and [Guidelines for Using Serialization](#).

```
public abstract class DlgcTest extends SipServlet implements Serializable
{
    private static final long serialVersionUID = -6161452986725649032L;
    //Since this value is a constant no need to serialize it; thus static is used
    protected static String dlgcDriverName = "com.dialogic.dlg309";
    //Note use of transient keyword: Factory, Listener, and Driver must be transient
    transient protected MsControlFactory mscFactory;
    transient protected DlgcSdpPortEventListener speListener;
    transient protected Driver dlgcDriver =null;

    private static Logger log = Logger.getLogger(DlgcTest.class);

    @Override
    public void init(ServletConfig cfg)
        throws ServletException
    {
        super.init(cfg);
        try
        {
            dlgcDriver = DriverManager.getDriver(dlgcDriverName);
            mscFactory = dlgcDriver.getFactory(null);
            speListener = new DlgcSdpPortEventListener();
        }
        catch (Exception e)
        {
            throw new ServletException(e);
        }
    }

    @Override
    public void doInvite(final SipServletRequest req)
        throws ServletException, IOException
    {
        log.info("doInvite");
        NetworkConnection networkConnection = null;
        SipSession sipSession = req.getSession();
        if (req.isInitial())
        {
            // We have a new call.

```

```

        try
        {
            MediaSession mediaSession = mscFactory.createMediaSession();
            networkConnection = mediaSession.createNetworkConnection(NetworkConnection.BASIC);
            sipSession.setAttribute("MEDIA_SESSION", mediaSession);
            sipSession.setAttribute("NETWORK_CONNECTION", networkConnection);

            //Even though this is not part of the best practices section
            //typically the application must store the UA SIP Session in the Media Session.
            //This is done because the UA SIP Session may be required inside a given Listener.
            //For example, the Listener may be required to send a Bye message to the UA.
            mediaSession.setAttribute("SIP_SESSION", sipSession);
            networkConnection.getSdpPortManager().addListener(speListener);
        }
        catch (MsControlException e)
        {
            req.createResponse(SipServletResponse.SC_SERVICE_UNAVAILABLE).send();
        }
    }
    .
    .
    .
}

```

## Example B

This example illustrates serialization best practices. The text in bold demonstrates the rule that all Listener classes must implement the serializable interface, as discussed in [Using Serialization](#).

```

private class PlayerEventListener implements MediaEventListener<PlayerEvent>, Serializable
{
    private static final long serialVersionUID = -50247033407045994L;
    @Override
    public void onEvent(PlayerEvent event)
    {
        log.info("PlayerEventListener::onEvent()");
        log.info("    EVENT TYPE : " + event.getEventType());
        ...
        //This shows how the UA sip session is used in the Listener to send the bye to the phone UA
        MediaSession mediaSession = event.getSource().getMediaSession();
        SipSession session = (SipSession) mediaSession.getAttribute("SIP_SESSION");
        log.debug("Calling BYE..Hanging Phone");
        SipServletRequest request = session.createRequest("BYE");
        try
        {
            request.send();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        .
        .
        .
    }
}

```

## 9. 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 *dlgmisc.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 through the OCCAS 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. *user1\_dlgmsec.properties*) is referencing the appropriate PowerMedia XMS and OCCAS IP address and ports.

## 10. Appendix: JSR 309 Connector Environment Setup

---

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

- [Installing and Configuring the OCCAS](#)
- [Installing the JSR 309 Connector](#)
- [Installing and Configuring the Test Servlets](#)
- [Running the Test Servlets](#)

For system requirements and supported platforms, see [System Requirements](#).

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

Steps to complete on OS level include:

- Install MySQL database
- Set up SSH for file configuration
- 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 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 JDK `.rpm` file from [www.oracle.com](http://www.oracle.com). For example, the following setup is based on `jdk-7u25-linux-x64.rpm` file.

Install the JDK `.rpm` file:

```
rpm -ivh jdk-7u25-linux-x64.rpm
```

Modify the `.bashrc` file and add the following line:

```
Export JAVA_HOME=/usr/java/jdk1.7.0_25
```

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.

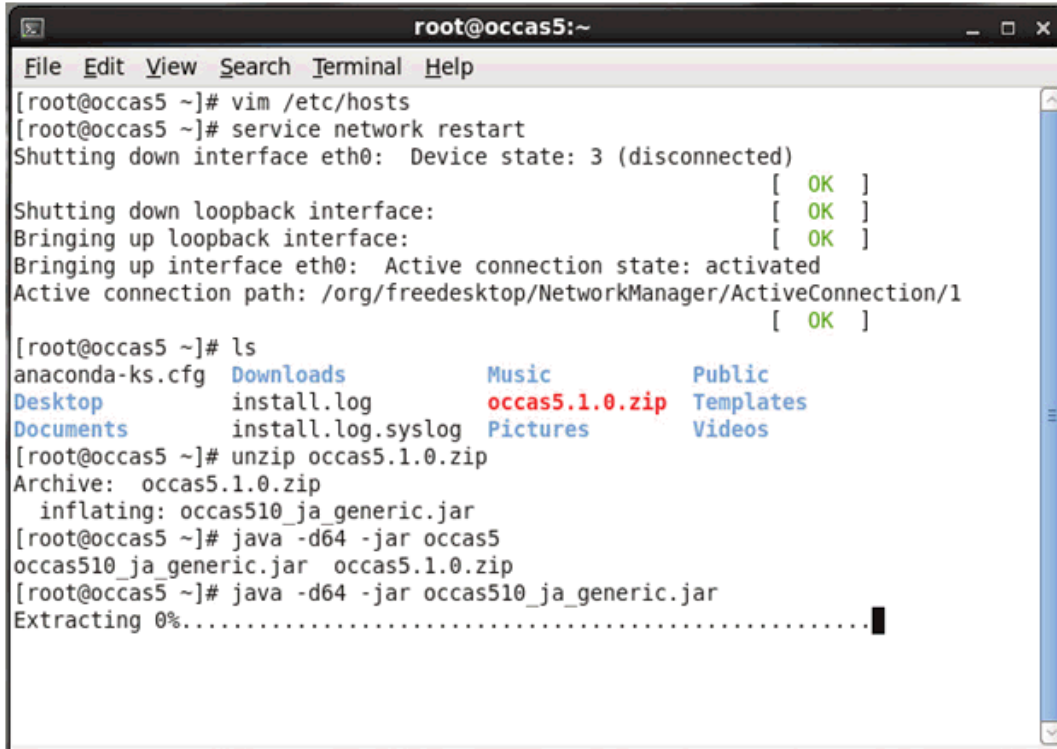
Unzip the `occas5.1.0.zip` file then proceed to [OCCAS Installation](#).

## OCCAS Installation

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

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

**Note:** You may need to change permissions in order for the file to be executable.



```

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 ]

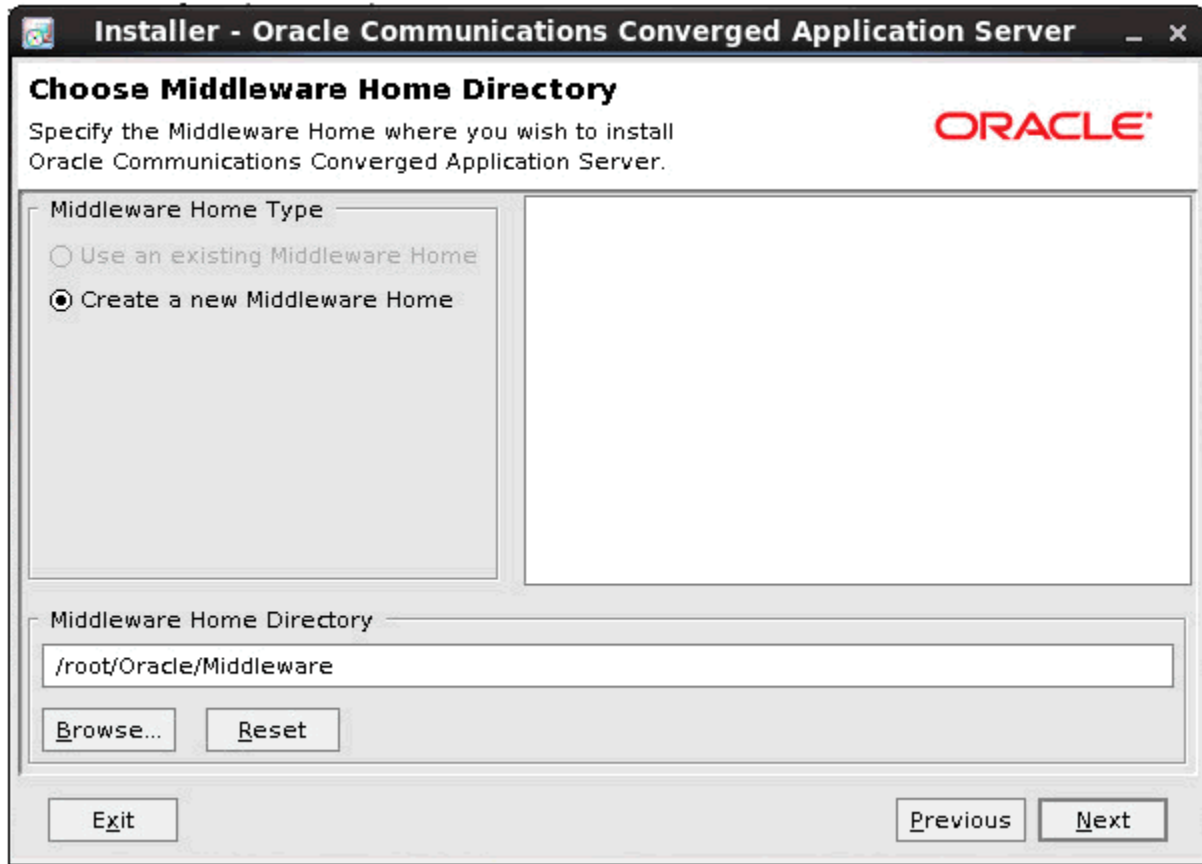
[root@occas5 ~]# ls
anaconda-ks.cfg  Downloads      Music          Public
Desktop          install.log    occas5.1.0.zip Templates
Documents        install.log.syslog Pictures        Videos

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

```



Click on **Next**.



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



The screenshot shows a window titled "Installer - Oracle Communications Converged Application Server". The main heading is "Register for Security Updates". Below the heading, it says: "Provide your email address to be informed of security issues, install the product, and initiate configuration manager. <http://www.oracle.com/support/policies.html>". The Oracle logo is in the top right corner. There is an "Email:" label followed by a text input field. Below the field, it says "Easier for you if you use your My Oracle Support email address/user...". There is a checked checkbox with the text "I wish to receive security updates via My Oracle Support". Below this, there is a "My Oracle Support Passw..." label followed by a password input field. At the bottom, there are three buttons: "Exit", "Previous", and "Next".

**Register for Security Updates**

Provide your email address to be informed of security issues, install the product, and initiate configuration manager. <http://www.oracle.com/support/policies.html>

Email:

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

☒ I wish to receive security updates via My Oracle Support

My Oracle Support Passw...

De-select security updates (unless you have an account with Oracle) then click on **Next**.



The screenshot shows the 'Register for Security Updates' window of the Oracle Communications Converged Application Server installer. The window has a title bar with the Oracle logo and the text 'Installer - Oracle Communications Converged Application Server'. The main content area has a light green header with the title 'Register for Security Updates' and the Oracle logo. Below the header, there is a text box for 'Email:' and a checkbox labeled 'I wish to receive security updates via My Oracle Support'. A 'My Oracle Support Passw...' text box is also present. At the bottom, there are three buttons: 'Exit', 'Previous', and 'Next'.

**Installer - Oracle Communications Converged Application Server**

**Register for Security Updates**

Provide your email address to be informed of security issues, install the product, and initiate configuration manager. <http://www.oracle.com/support/policies.html>

Email:

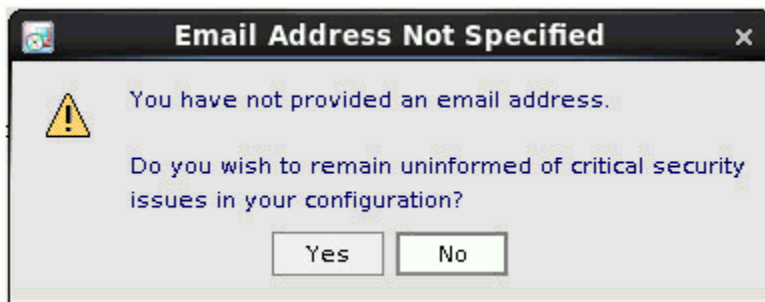
Easier for you if you use your My Oracle Support email address/user...

☒ I wish to receive security updates via My Oracle Support

My Oracle Support Passw...


**Exit** **Previous** **Next**

Click on **Next**.



The screenshot shows a dialog box titled 'Email Address Not Specified'. It contains a yellow warning icon, the text 'You have not provided an email address.', and a question 'Do you wish to remain uninformed of critical security issues in your configuration?'. At the bottom, there are two buttons: 'Yes' and 'No'.

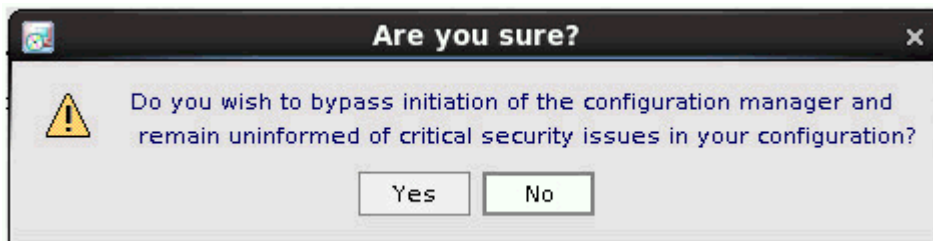
**Email Address Not Specified**

 You have not provided an email address.

Do you wish to remain uninformed of critical security issues in your configuration?


**Yes** **No**

Click on **Yes**.



The screenshot shows a dialog box titled 'Are you sure?'. It contains a yellow warning icon, the text 'Do you wish to bypass initiation of the configuration manager and remain uninformed of critical security issues in your configuration?', and two buttons: 'Yes' and 'No'.

**Are you sure?**

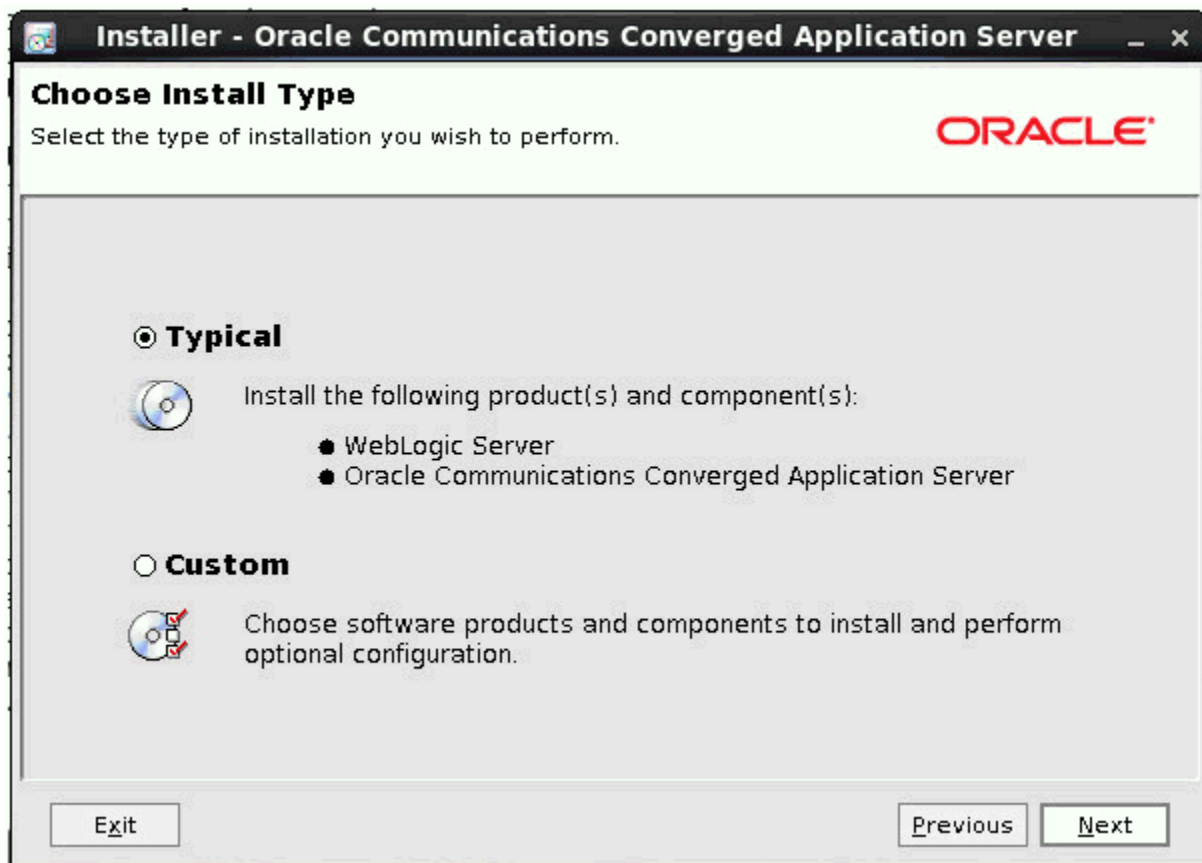
 Do you wish to bypass initiation of the configuration manager and remain uninformed of critical security issues in your configuration?

**Yes** **No**

Click on **Yes**.



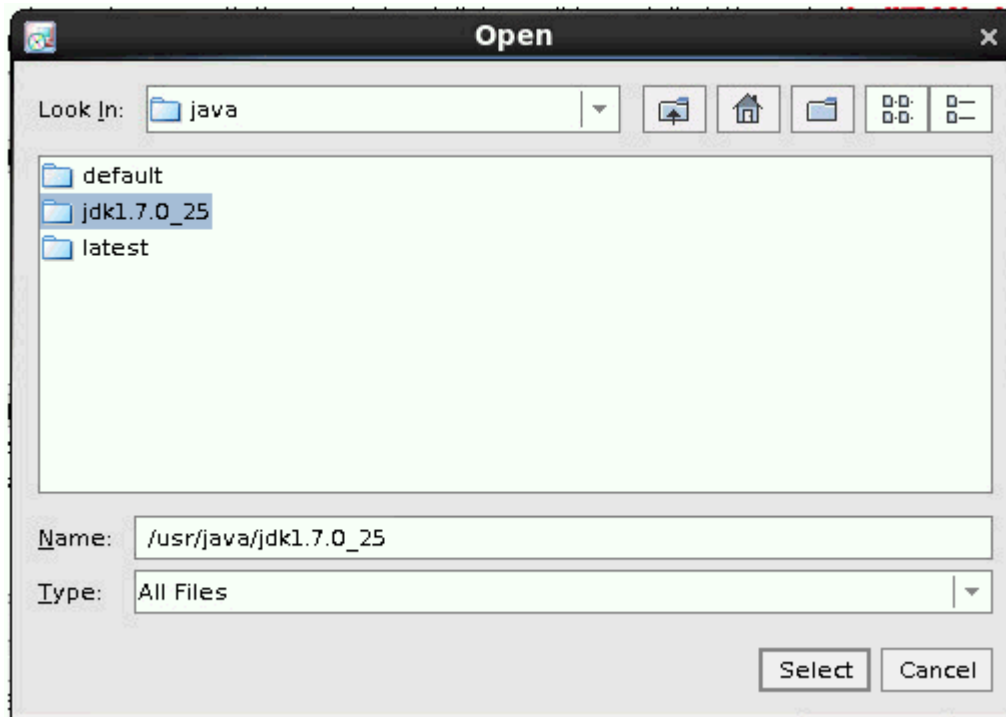
Select **I wish to remain uninformed...** then click on **Continue**.



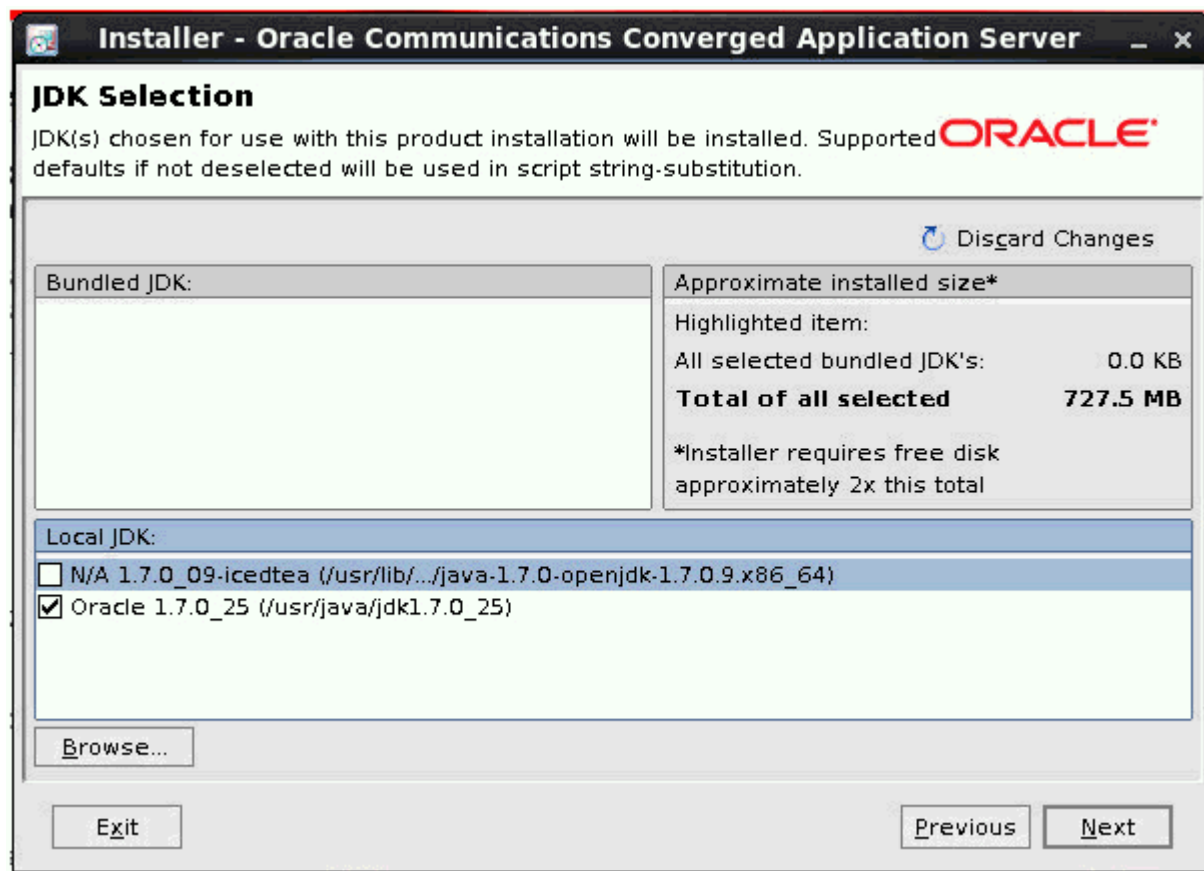
Select **Typical** then click on **Next**.



Browse to previously installed **jdk1.7.0\_25** directory.



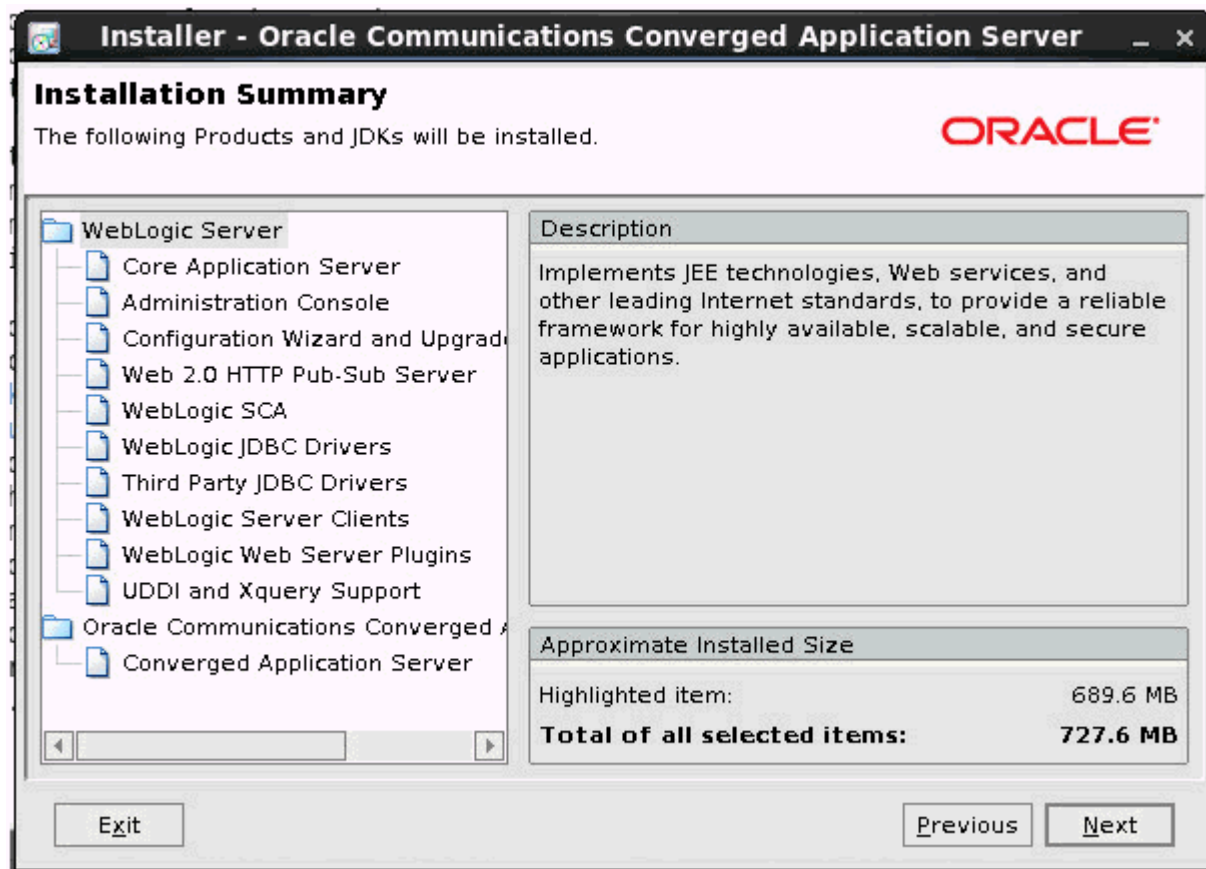
Select **jdk1.7.0\_25** directory then click on **Select**.



Make sure the selected **jdk1.7.0\_25** directory is the only JDK checked then click on **Next**.



Click on **Next**.



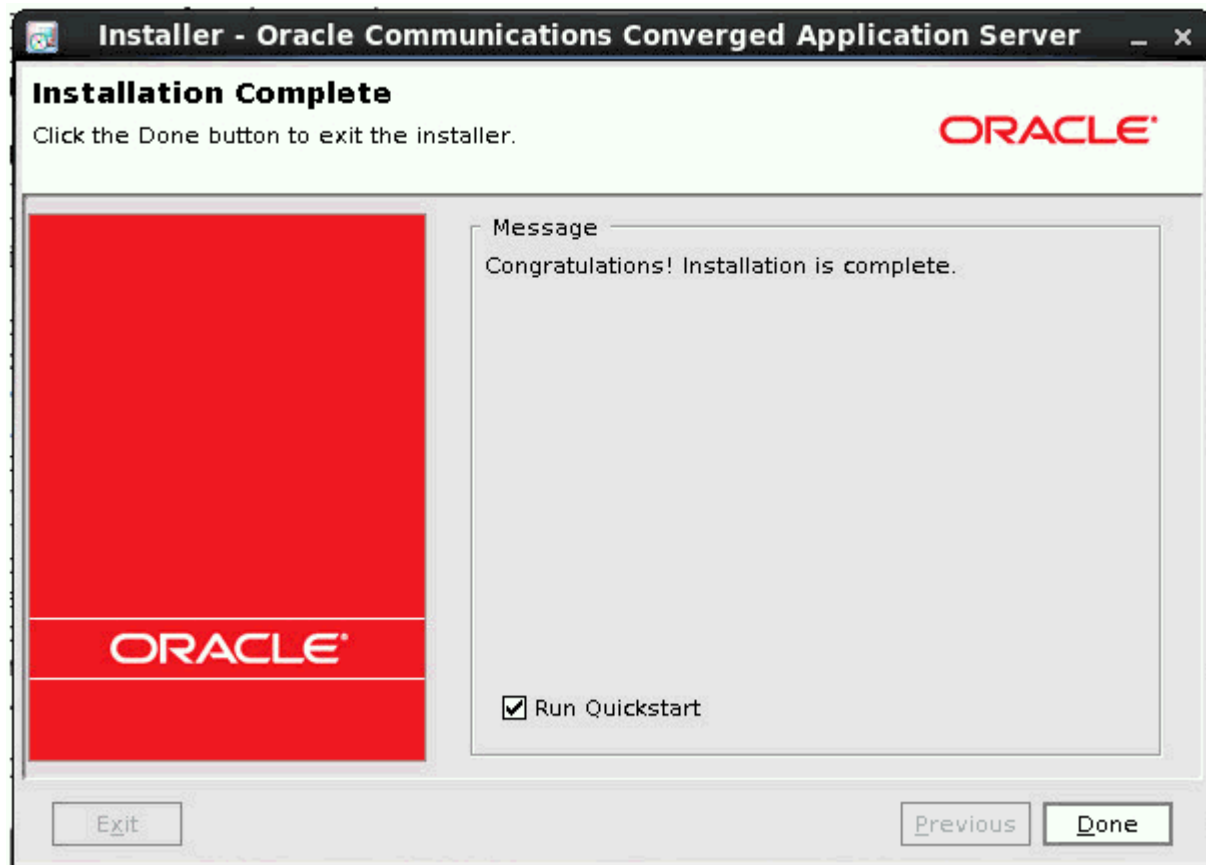
Click on **Next**.



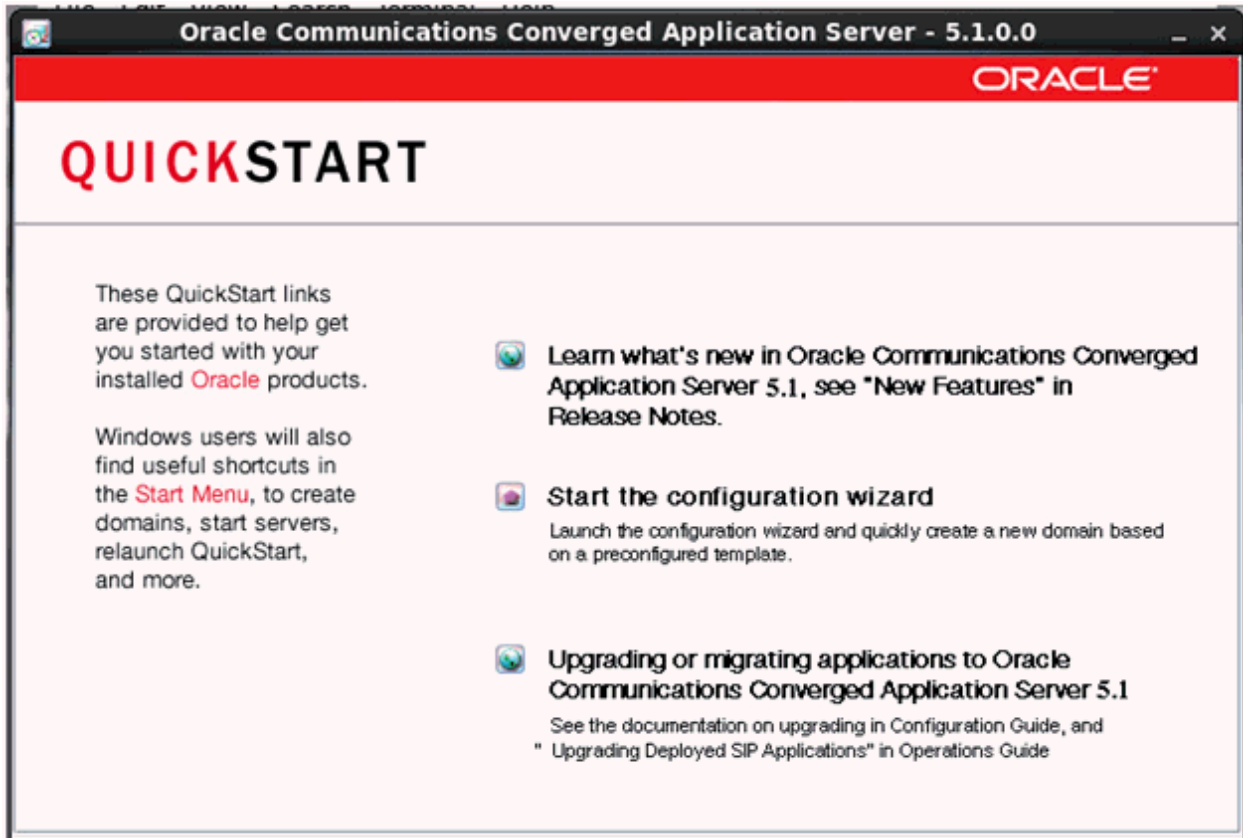
Click on **Next**.

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

## OCCAS Configuration



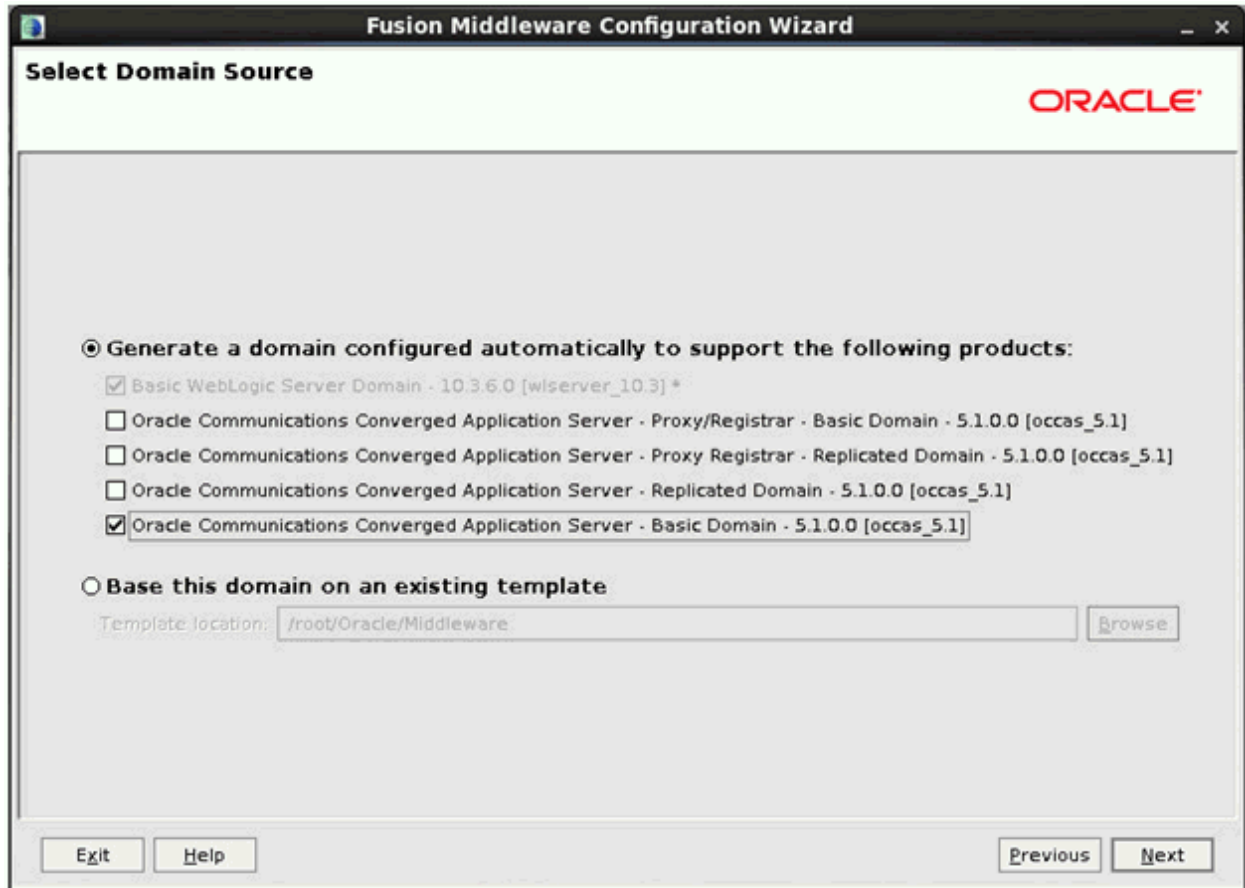
Select **Run Quickstart** then click on **Done**.



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**.



The image shows a screenshot of the 'Fusion Middleware Configuration Wizard' window. The title bar reads 'Fusion Middleware Configuration Wizard'. The main heading is 'Specify Domain Name and Location'. The Oracle logo is in the top right corner. The instruction 'Enter the name and location for the domain:' is centered. Below this, there are two input fields: 'Domain name:' with the text 'base\_domain' and 'Domain location:' with the text '/root/Oracle/Middleware/user\_projects/domains'. A 'Browse' button is next to the domain location field. At the bottom, there are four buttons: 'Exit', 'Help', 'Previous', and 'Next'.

**Fusion Middleware Configuration Wizard**

**Specify Domain Name and Location**

**Enter the name and location for the domain:**

Domain name:

Domain location:

Click on **Next**.

Fusion Middleware Configuration Wizard

Configure Administrator User Name and Password

ORACLE

[Discard Changes](#)

\*Name: weblogic

\*User password: \*\*\*\*\*

\*Confirm user password: \*\*\*\*\*

Description: This user is the default administrator.

Exit Help Previous Next

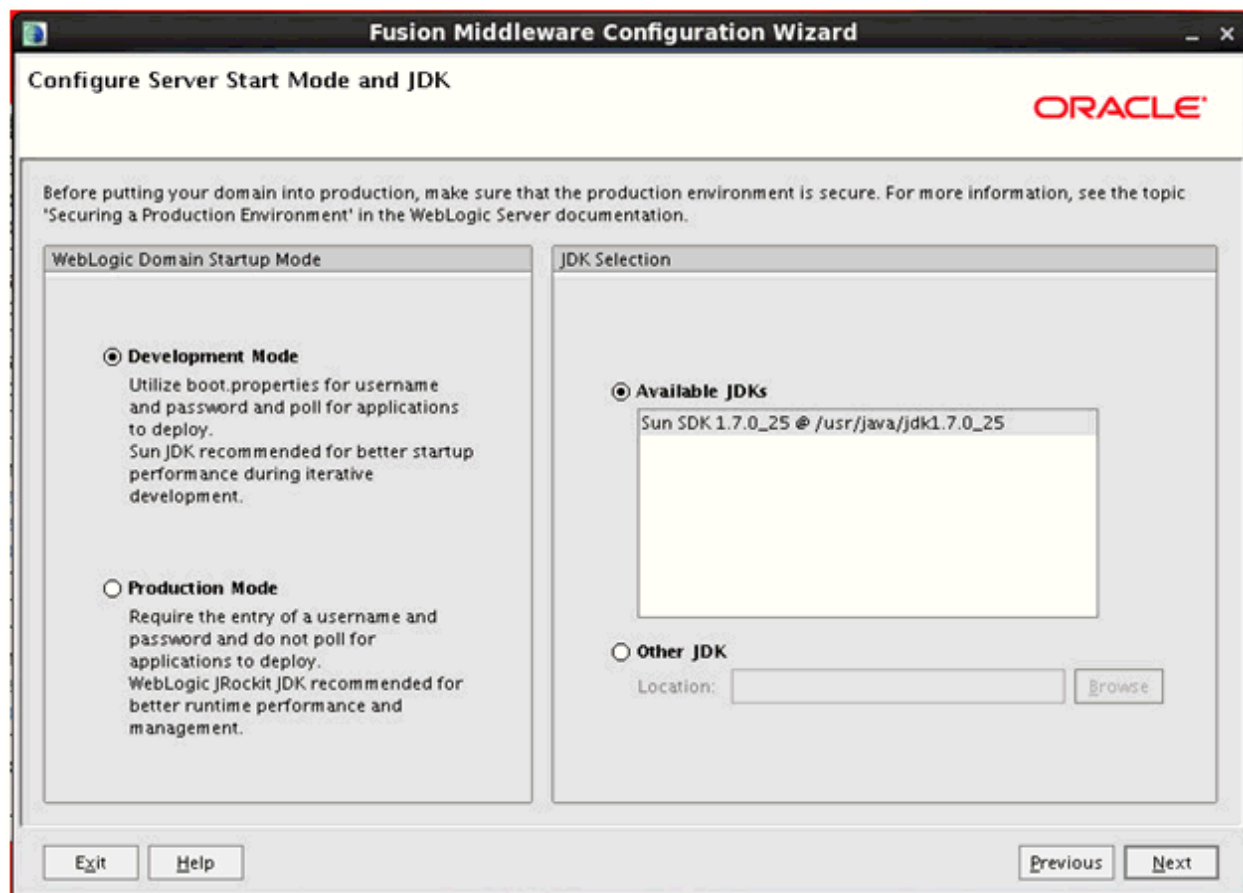
Specify **Name** and **User password** then click on Next.

The following is used as an example:

**Name:** weblogic

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

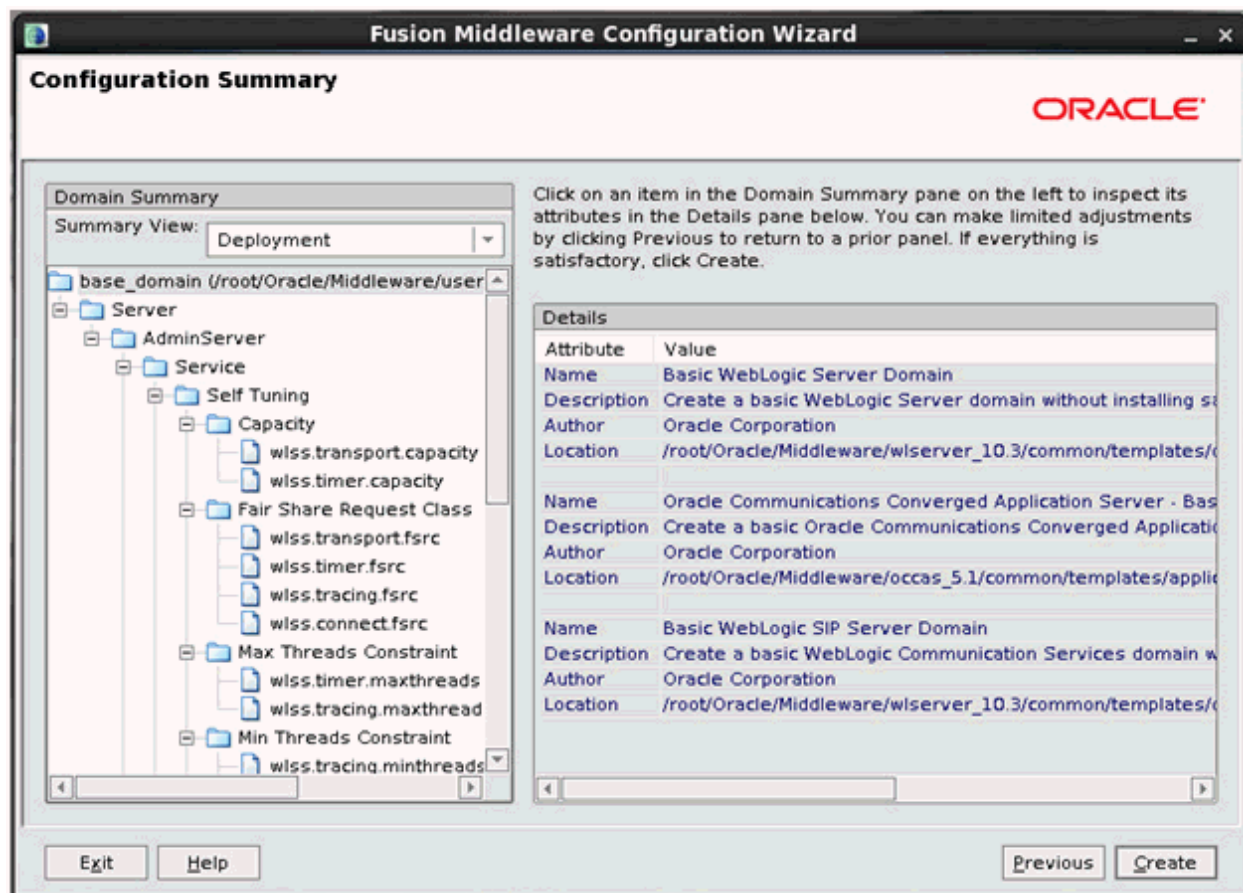
**Note:** A strong password is required.



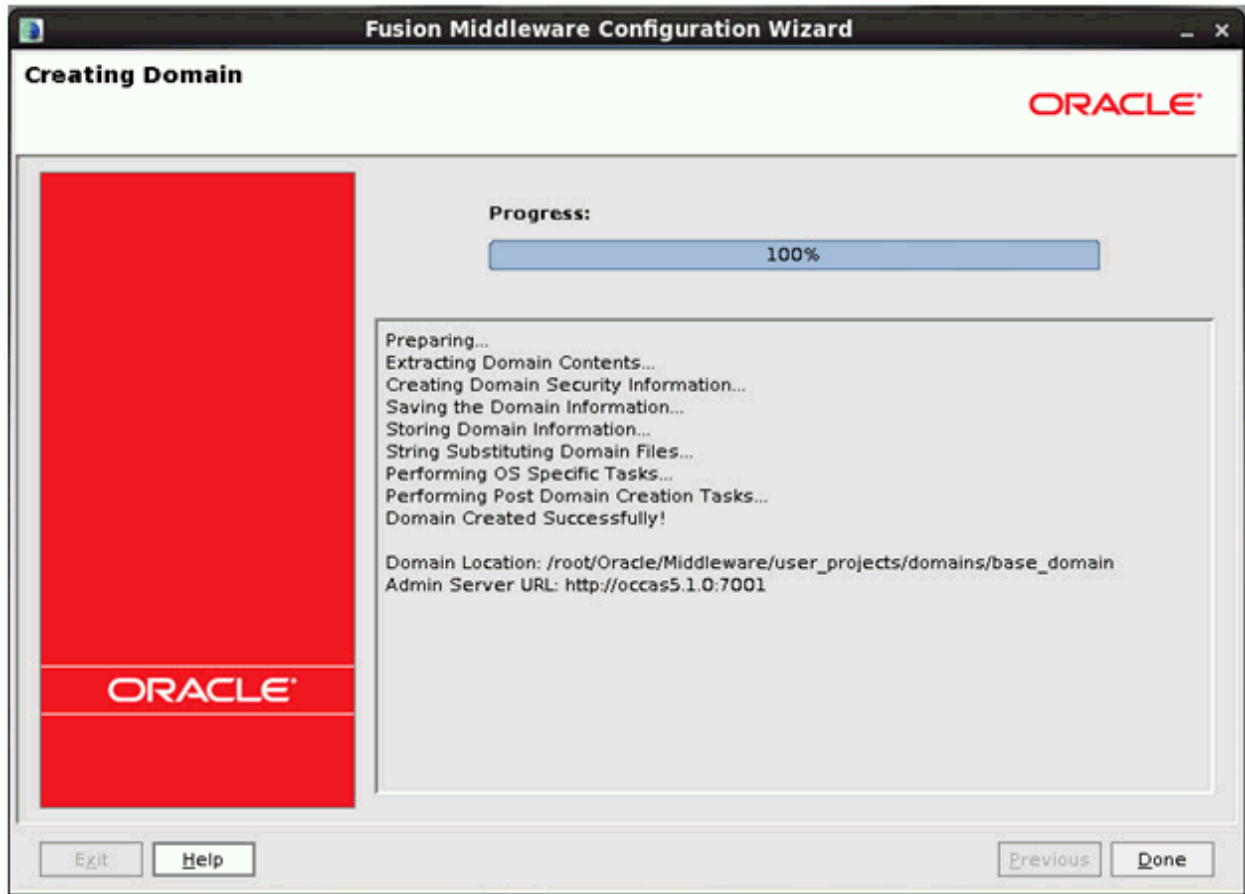
Select **Development Mode** then click on **Next**.



Click on **Next**.



Click on **Create**.



The OCCAS installation and configuration is 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
```



A terminal window titled "root@occas5:~/Oracle/Middleware/user\_projects/domains/base\_domain/bin". The window shows the command "pwd" being executed, which returns the path "/root/Oracle/Middleware/user\_projects/domains/base\_domain/bin". The prompt is "[root@occas5 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
```



A terminal window titled "root@occas5:~/Oracle/Middleware/user\_projects/domains/base\_domain/bin". The window shows the command "pwd" being executed, which returns the path "/root/Oracle/Middleware/user\_projects/domains/base\_domain/bin". Then, the command "./startWebLogic.sh" is entered, and the prompt is "[root@occas5 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]# ./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=AdminServ
er -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/wls
erver_10.3 -Dwls.home=/root/Oracle/Middleware/wlserver_10.3/server -Dweblogic.home=/root/Oracle/Middle
ware/wlserver_10.3/server -Dweblogic.management.discover=true -Dwlw.iterativeDev=false -Dwlw.testCo
nsole=false -Dwlw.logErrorsToConsole=false -Dweblogic.ext.dirs=/root/Oracle/Middleware/patch_wls1036/p
rofiles/default/sysexst_manifest_classpath:/root/Oracle/Middleware/patch_occas510/profiles/default/syse
xt_manifest_classpath weblogic.Server
<Jun 4, 2013 6:58:29 PM EDT> <Info> <Security> <BEA-090905> <Disabling CryptoJ JCE Provider self-integ
rity check for better startup performance. To enable this check, specify -Dweblogic.security.allowCryp
toJDefaultJCEVerification=true>
<Jun 4, 2013 6:58:29 PM EDT> <Info> <Security> <BEA-090906> <Changing the default Random Number Genera
tor in RSA CryptoJ from ECDRBG to FIPS186PRNG. To disable this change, specify -Dweblogic.security.all
owCryptoJDefaultPRNG=true>
<Jun 4, 2013 6:58:29 PM EDT> <Notice> <WebLogicServer> <BEA-000395> <Following extensions directory co
ntents 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.act
ivator, 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 OpenJ
DK 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)

```

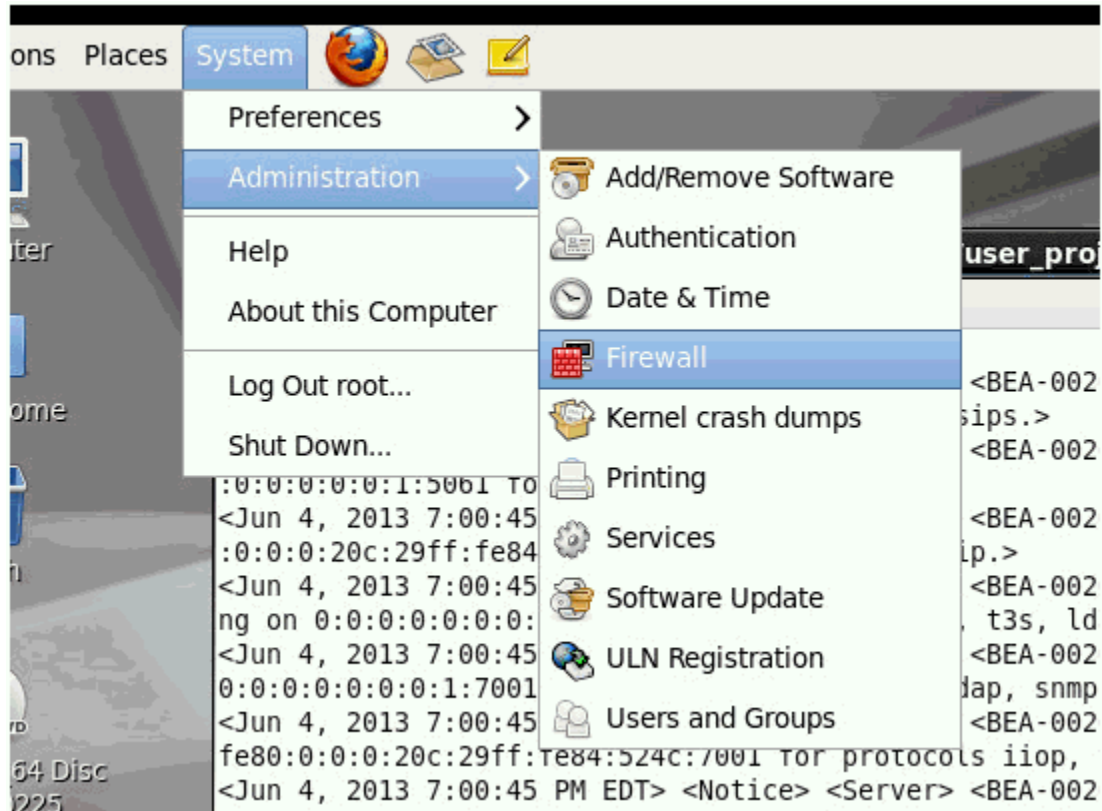
root@occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
File Edit View Search Terminal Help
0:0:0:0:0:1:5060 for protocols sip.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[2]" is now listening on fe8
0:0:0:0:20c:29ff:fe84:524c:5061 for protocols sips.>
<Jun 4, 2013 7:00:45 PM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[4]" is now listening on 0:0
:0:0:0:0:1:5061 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 listeni
ng on 0:0:0:0:0:0:0:1:7002 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
0:0:0:0:0:0:0:1:7001 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 listeni
ng 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 listeni
ng 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 "Ad
minServer" 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 (Tr
ansport 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>

```

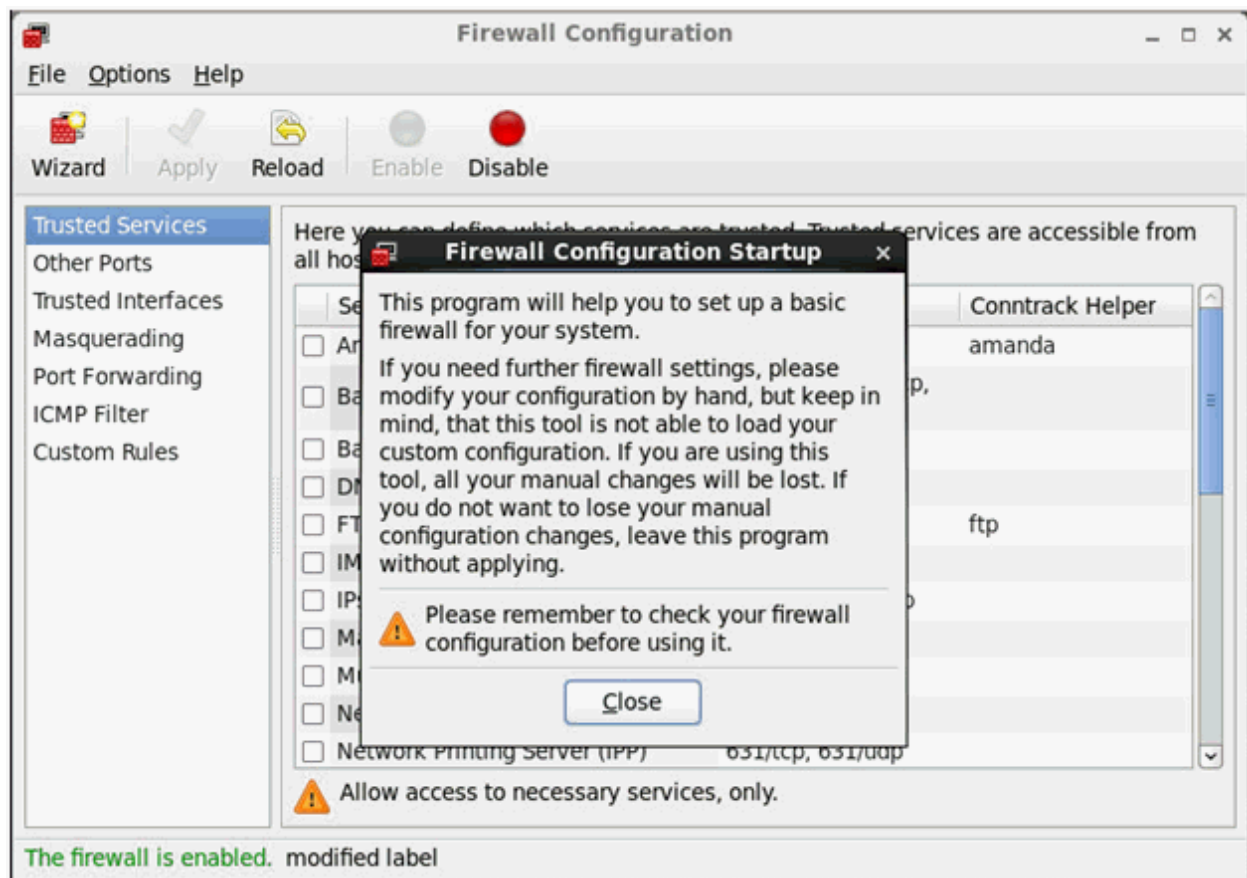
To verify that OCCAS is started, check if **<Server started in RUNNING mode>** is displayed.

## Firewall Configuration

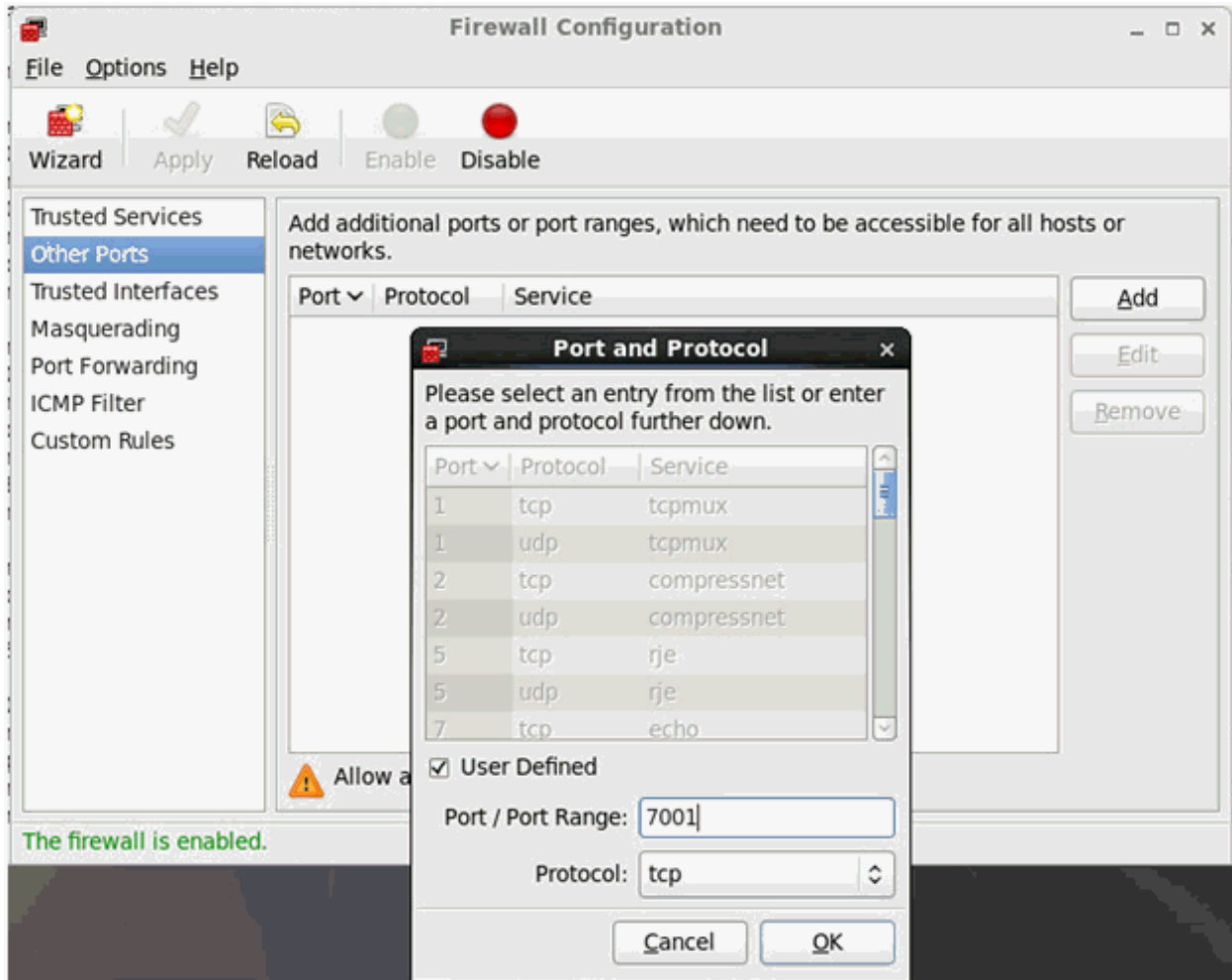
Enable port 7001/tcp, 5060/udp, and 5061/udp in the Linux firewall.



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

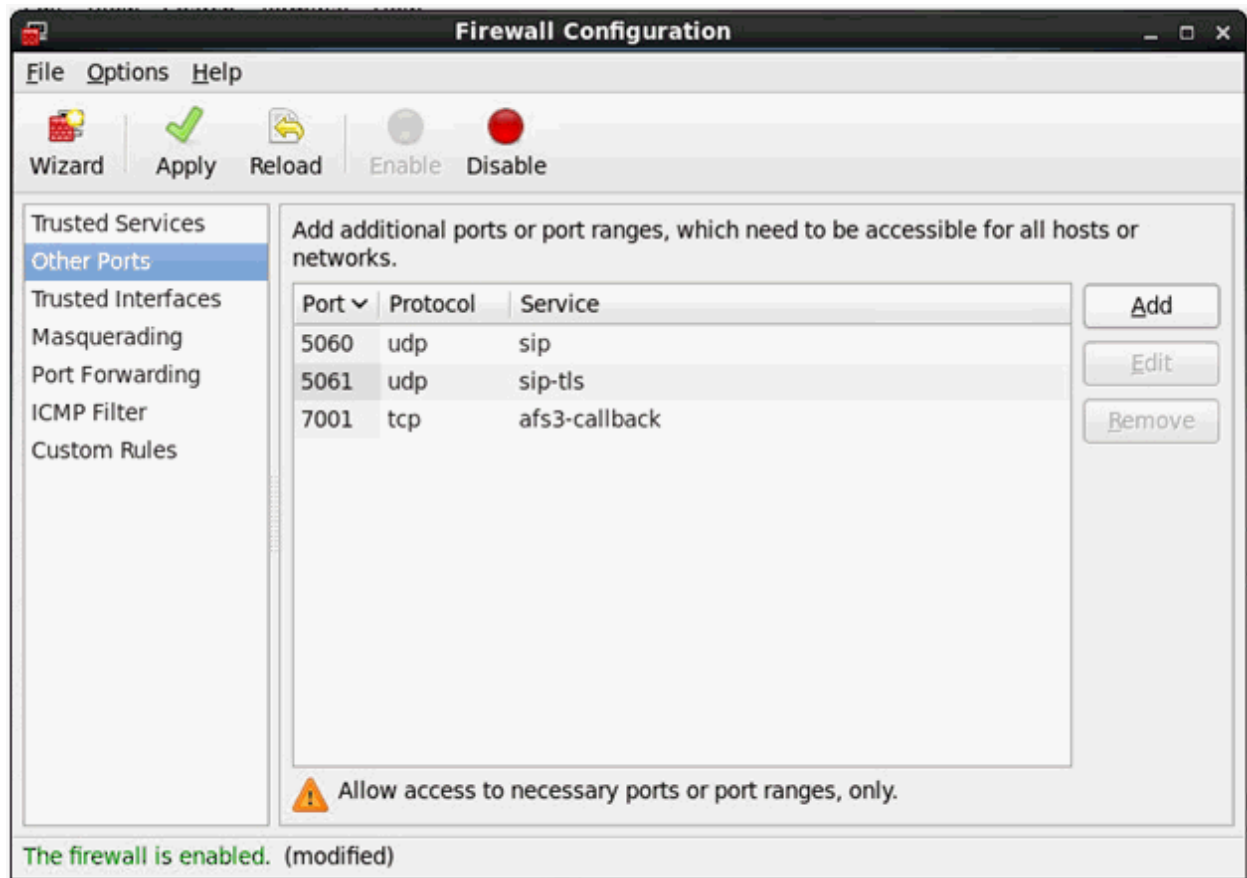


Click on **Close**.

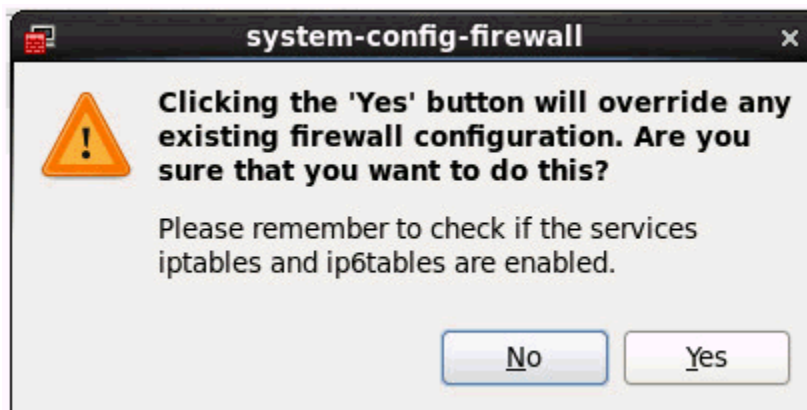


Select **Other Ports** then click **Add**. Check the **User Defined** box, enter **7001** for **Port/Port Range**, choose **tcp** for **Protocol**, and then click **OK**.

Repeat the steps to add **5060** and **5061** for **Port/Port Range** but with **udp** as **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://<OCCAS5.1.0-IP-ADDRESS>:7001/console>

The screenshot shows the Oracle WebLogic Server Administration Console. The left sidebar displays the Domain Structure, with 'Deployments' selected under 'base\_domain'. The main area shows the 'Summary of Deployments' page, which includes a table of installed applications and modules.

Name	State	Health	Type	Deployment Order
java-rt(1.1.1.0)	Active		Library	100
jsr311-api(1.1.1.1.0)	Active		Library	100
merp-connector-ra	Active	OK	Resource Adapter	100
ycdb-webglue(5.1.0.0.0,5.1.0.0.0)	Active		Library	100
yaff(1.0.1.0)	Active		Library	100

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

## Installing the JSR 309 Connector

Follow this procedure to get the application (WAR file) in an OCCAS environment to correctly load the JSR 309 Connector (*dlgmsc.jar*).

The JSR 309 Connector supports PowerMedia XMS via the MSML protocol.

### Step 1

Extract the *JSR309\_MSC3\_BuildXX.tar* distribution package for this release which will create a <Release Package> directory.

**Note:** <Release Package> refers to the *.tar* distribution package provided for this release.

Copy all the *.jar* files from <Release Package>/DlgcJSR309/lib directory to the OCCAS <Domain Location>/lib directory.

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

JAR Files	Description
<i>dlgmsc.jar</i> <i>jain-sip-sdp-1.2.91.jar</i> <i>jsr173_1.0_api.jar</i> <i>log4j-1.2.15.jar</i> <i>msmltypes.jar</i> <i>slf4j-api-1.7.2.jar</i> <i>slf4j-log4j12-1.7.2.jar</i> <i>xbean.jar</i>	JSR 309 Connector implementation for PowerMedia XMS and required third-party software libraries.

### Step 2

Set up the properties files.

JSR 309 Connector uses the following properties files:

- *dlgcJSR309.properties* - used to configure IP addresses and ports for the OCCAS environment using JSR 309 Connector and dedicated PowerMedia XMS platform.
- *log4j.properties* - used for logging using Simple Logging Facade framework implementation of log4j.

**Note:** You can configure logging in the *log4j.properties* file. By default, logging is configured to use INFO level and output to the Console as well as to a *dlgmsc.log* file.

Follow these steps to set up the properties file:

1. Copy the *dlgcJSR309.properties* and *log4j.properties* files from <Release Package>/DlgcJSR309/properties directory to the OCCAS <Domain Location>/config directory.
2. Edit the *dlgcJSR309.properties* file according to your OCCAS and PowerMedia XMS configuration. The changes will include the OCCAS IP address and SipServlet container port running the JSR 309 Connector as well as the PowerMedia XMS IP address and port:

# JSR 309 Connector address information (typically same as the SipServlet container)

```
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5060
```

## # PowerMedia XMS address information

```
mediaserver.1.sip.address=xxx.xxx.xxx.xxx
mediaserver.1.sip.port=5060
```

3. Edit the <Domain Location>/bin/startWebLogic.sh OCCAS startup script. Look for the following line:

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

## Add the following lines directly after:

```
export DIALOGIC_DEMO_PROPERTY_FILE=${DOMAIN_HOME}/applications/dlgc_demos.properties
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/dlgcJSR309.properties
LOG4J_OPTIONS="-Dlog4j.configuration=${DOMAIN_HOME}/config/log4j.properties"
XQUERYPATH=/root/Oracle/Middleware/modules/features/weblogic.server.modules.xquery_10.3.6
.jar
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/serve
r/lib/jsr309-descriptor-binding.jar:${XQUERYPATH}"
```

4. Add the following line in OCCAS startup script to enable some of the relevant items and to disable serialization (highlighted in bold):

```
${LOG4J_OPTIONS} -Dlog4j.debug -Dwlss.local.serialization=false
```

## from:

```
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
    echo "Starting WLS with line:"
    echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME}
-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} -
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} -
Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS}
${PROXY_SETTINGS} ${SERVER_CLASS} > "${WLS_REDIRECT_LOG}" 2>&1
fi
```

## to:

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

For more details on disabling serialization, refer to [Disabling Serialization](#).

5. Save the startup script file then restart the WebLogic Server.

## Installing and Configuring the Test Servlets

Make sure that desired application exists in <Domain Location>/application.

Access the **Administration Console** at:

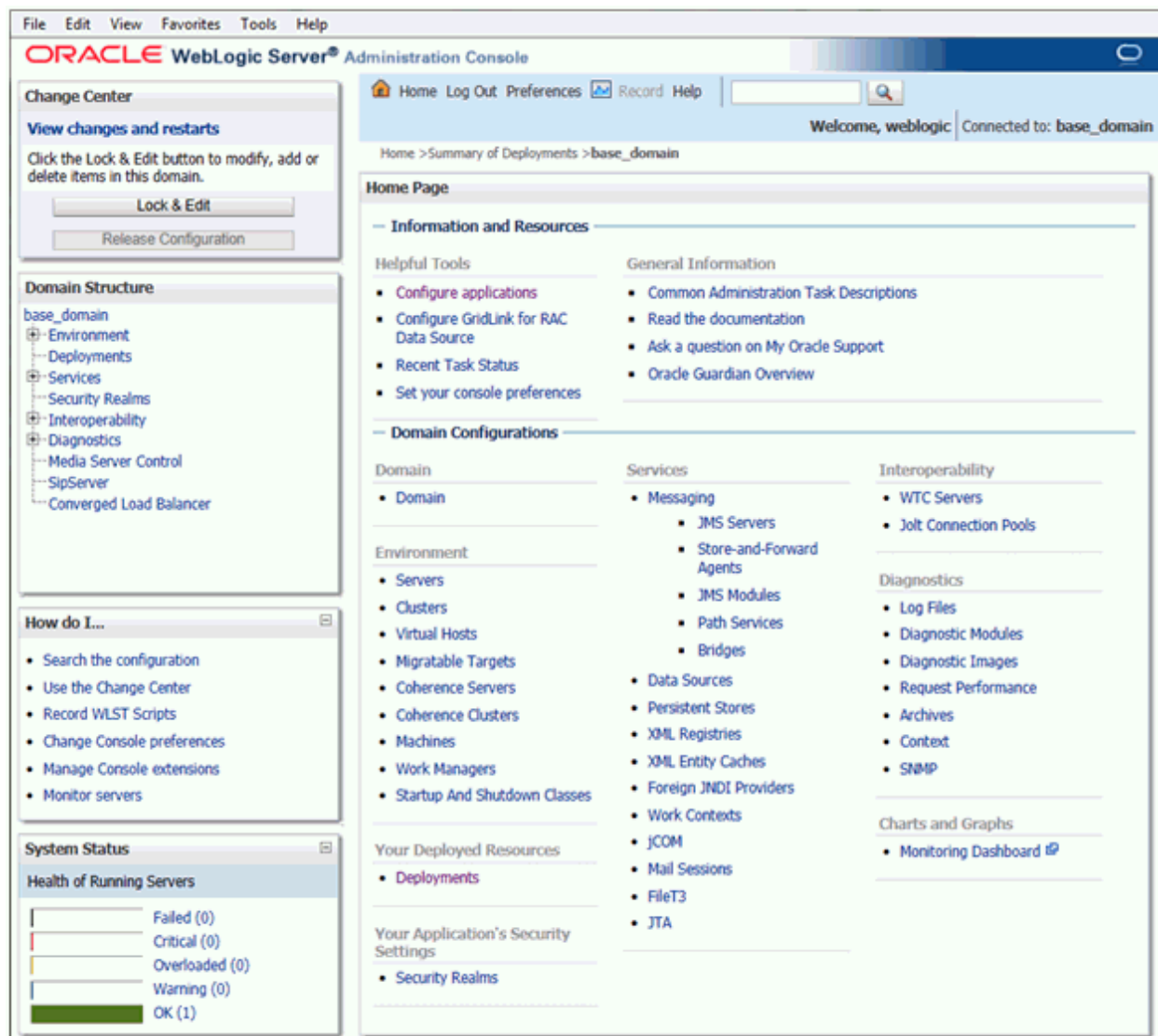
http://<OCCAS5.1.0-IP-ADDRESS>:7001/console

Use the username/password as set during configuration.

The following is used as an example:

**Name:** weblogic

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



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

The screenshot shows the Oracle WebLogic Server Administration Console. The left sidebar contains a 'Change Center' section with 'View changes and restarts' and buttons for 'Lock & Edit' and 'Release Configuration'. Below this is the 'Domain Structure' tree, which includes 'base\_domain' and its sub-components like 'Environment', 'Deployments', 'Services', etc. The main content area is titled 'Summary of Deployments' and features a 'Control' tab. It contains a table of installed applications and modules.

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 as shown 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 activating 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 on the Control tab.

To install a new application or module for deployment, click on 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

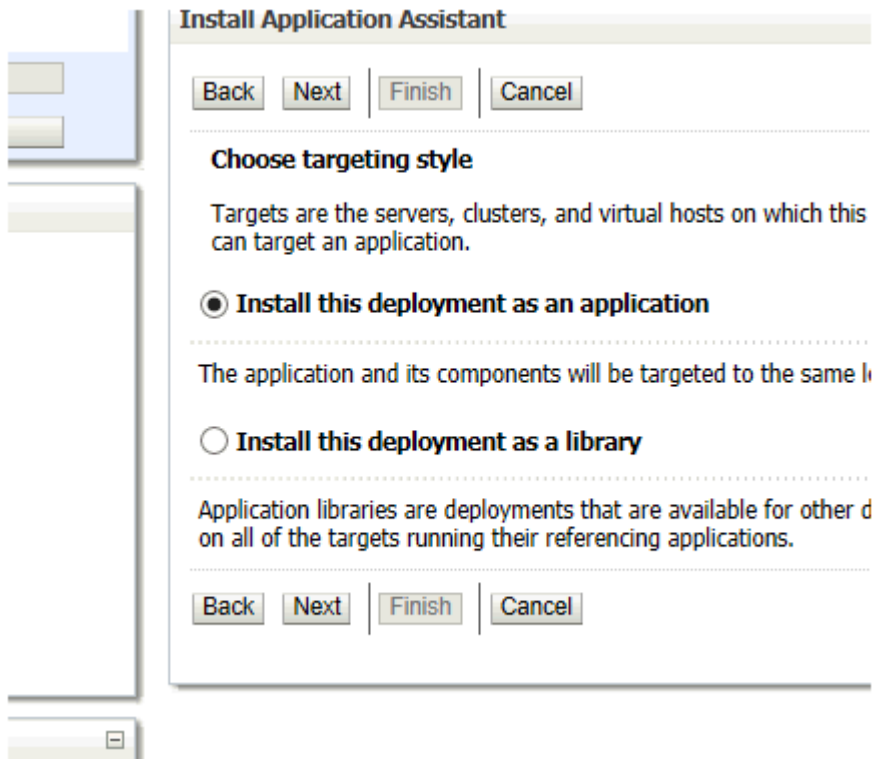
Current Location: 146.152.121.155 / root / Oracle / Middleware / user\_projects / domains / base\_domain / applications

☐ dlgmisc\_tests.war

Back Next Finish Cancel

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

Select **dlgmisc\_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**

BackNextFinishCancel

**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



BackNextFinishCancel

Click on **Finish**.

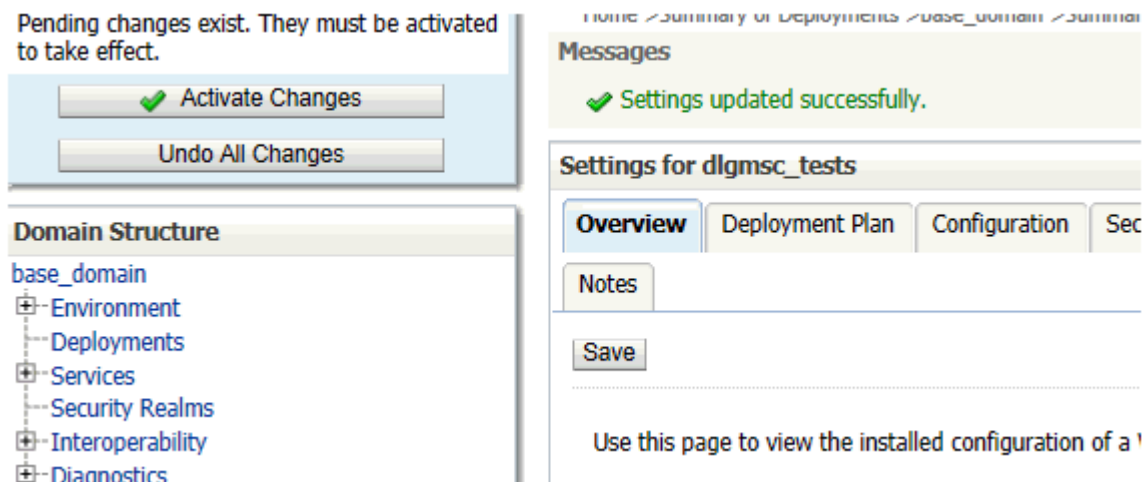
**Settings for dlghmc\_tests**

**Overview** | Deployment Plan | Configuration | Security | Targets | Control | Testing | Mon

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

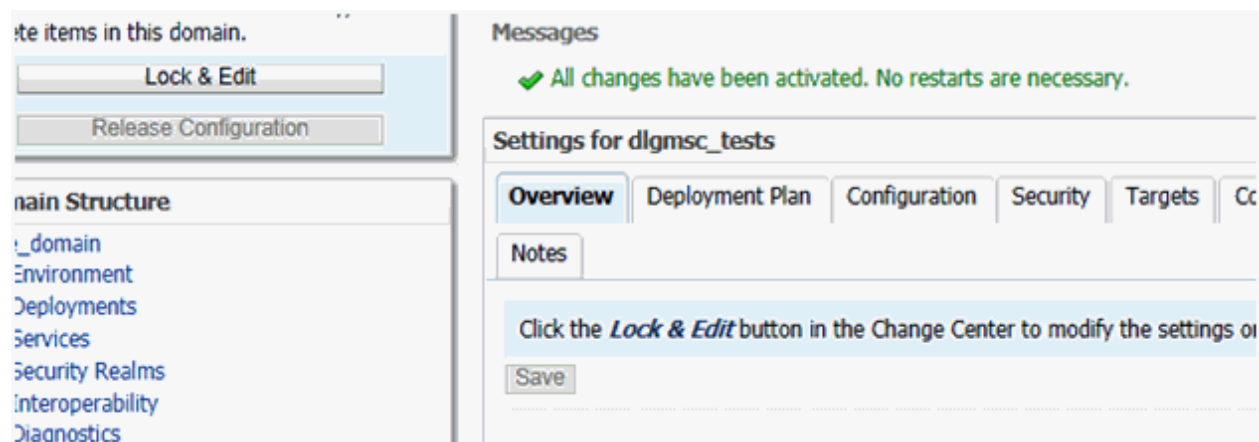
<b>Name:</b>	dlghmc_tests	The name of this <a href="#">applic</a> <a href="#">Info...</a>
<b>Context Root:</b>	/dlghmc_tests	The specific path at whi found by a servlet. <a href="#">Mc</a>
<b>Path:</b>	/ root/ Oracle/ Middleware/ user_projects/ domains/ base_domain/ applications/ dlghmc_tests. war	The path to the source the Administration Serv
<b>Deployment Plan:</b>	(no plan specified)	The path to the deployr 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 otl server, during startup.
 <b>Deployment Principal Name:</b>	<input type="text"/>	A string value that indic be used when deploying startup and shutdown. to set the environment

Click on **Save**.

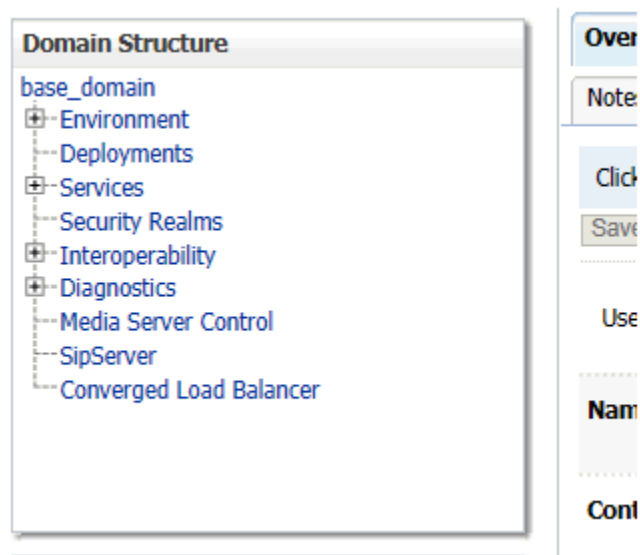


Click on **Activate Changes** once the "Settings updated successfully" message appears.

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



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



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

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 description of the page and a "Customize this table" link. A table titled "Deployments" lists the installed applications and modules. The table has columns for Name, State, Health, Type, and Deployment Order. The first row shows "dlgmisc\_tests" with a State of "Prepared" and Health of "OK". Other rows show various libraries and connectors.

Name	State	Health	Type	Deployment Order
dlgmisc_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
scib-webglue(5.1.0.0.0,5.1.0.0.0)	Active		Library	100
sft(1.0,1.0)	Active		Library	100

WebLogic Server Version: 10.3.6.0  
Copyright © 1996, 2011, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

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

**Deployments**

Install Update Delete Start Stop Showing 1 to

<input type="checkbox"/>	Name	State	Health	Type
<input checked="" type="checkbox"/>	dlgmsc_tests	Prepared	OK	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	OK	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 *dlgmsc\_tests.war* application and click on **Start** then **Servicing all requests**.

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

Home > Summary of Deployments > base\_domain > Summary of Deployments > dlgmsc\_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.

- dlgmsc\_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.

## Running the Test Servlets

Sample test servlets use custom demo prompts which need to be installed on dedicated PowerMedia XMS platform for this configuration. You can locate and install the demo prompts by performing the following:

1. Copy all the *.tar* files inside the <Release Package>/DlgcJSR309DemoPrompts directory to the PowerMedia XMS machine under `/var/lib/xms/media/en_US/verification` directory.
2. Untar each of the media files using the command; `tar -xvf <Filename>.tar`.

To verify that your installation is successful, you can dial into OCCAS and run a simple demo included with the JSR 309 Connector which will play a file.

To test the application, dial the following:

```
DlgcPlayerDemo@<OCCAS5.1.0-IP-ADDRESS>
```

If it is successful, you will hear a "Please contact your service provider" prompt.