



JSR 309 Connector Software for Dialogic® PowerMedia™ XMS

Quick Start Guide

Copyright and Legal Notice

Copyright © 2010-2013 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.

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

| | |
|---|-----------|
| 1. Welcome | 7 |
| Assumptions | 7 |
| Related Information | 7 |
| 2. Overview..... | 8 |
| Terminology | 8 |
| JSR 309 Media Server Control API | 8 |
| JSR 309 Connector | 9 |
| System Overview | 9 |
| Application Programming Model..... | 9 |
| 3. Supported Features..... | 10 |
| Limitations | 10 |
| Supported Configurations | 11 |
| 4. System Requirements | 12 |
| Hardware and Software Requirements | 12 |
| Supported Platforms | 12 |
| 5. Contents of the Distribution | 13 |
| Distributed Files | 13 |
| Demo Prompts | 14 |
| 6. Installation and Configuration | 15 |
| Preparing the J2EE Converged Application Server | 15 |
| Installing the JSR 309 Connector | 15 |
| Configuring Logging..... | 17 |
| 7. Test Servlets | 18 |
| Test Servlets Overview..... | 18 |
| Installing the Demo Prompts..... | 18 |
| Running the Test Servlets..... | 18 |
| DlgcPlayerTest | 18 |
| DlgcDtmfPromptAndCollectTest | 19 |
| DlgcRecorderTest | 19 |
| DlgcDtmfAsyncTest..... | 19 |
| Conference Demos..... | 20 |
| JMCConferenceServlet..... | 20 |
| DialogicQuickConferenceDemo | 21 |
| DlgcReferenceConferenceDemo | 22 |
| DialogicBridgeConference | 22 |
| DlgcEarlyMediaBridgeDemo..... | 23 |
| 8. Developer Guidelines | 26 |
| General Guidelines | 26 |
| Early Media Guidelines | 26 |
| Redundant Media Servers Guidelines | 26 |
| Configuring Network Time Protocol (NTP) for Accurate SIP Timers | 27 |
| Configuring the Application Server..... | 27 |
| Using Serialization | 27 |
| Disabling Serialization | 27 |
| Guidelines for Using Serialization | 28 |
| Configuration for a Distributed Environment | 29 |
| Example Code | 29 |

| | |
|--|-----------|
| Example A | 29 |
| Example B | 30 |
| 9. Troubleshooting | 31 |
| Logging..... | 31 |
| SIP Errors..... | 31 |
| 10. Appendix: JSR 309 Connector Environment Setup | 32 |
| Installing the OCCAS..... | 32 |
| Pre-Installation Setup | 32 |
| Database Setup..... | 33 |
| Database Creation | 33 |
| OCCAS .bin Installation | 33 |
| OCCAS Startup..... | 49 |
| Installing the JSR 309 Connector..... | 52 |
| Installing and Configuring the Test Servlets | 55 |
| Running the Test Servlets..... | 63 |

Revision History

| Revision | Release Date | Notes |
|-------------|---------------|---|
| 05-2708-004 | April 2013 | <p>Supported Features:</p> <ul style="list-style-type: none"> Added new supported configurations in Supported Configurations table. <p>Test Servlets:</p> <ul style="list-style-type: none"> Added new section for <code>DlgcReferenceConferenceDemo</code>. |
| 05-2708-003 | March 2013 | <p>Supported Features:</p> <ul style="list-style-type: none"> Added new video file playback and video conference features. |
| 05-2708-002 | February 2013 | <p>Contents of the Distribution and Installation and Configuration:</p> <ul style="list-style-type: none"> Added new files to support Simple Logging Facade framework. |
| 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"> Renamed Dialogic® PowerMedia™ Media Server Connector (PowerMedia MSC) to JSR 309 Connector Software for Dialogic® PowerMedia™ XMS (JSR 309 Connector). Integrated content from Release Notes into this Quick Start Guide. <p>Supported Features:</p> <ul style="list-style-type: none"> Added new bridge conference feature. <p>System Requirements:</p> <ul style="list-style-type: none"> Added new section for hardware and software requirements along with supported platforms. <p>Installation and Configuration:</p> <ul style="list-style-type: none"> Added new content for installation and configuration. <p>Test Servlets:</p> <ul style="list-style-type: none"> Added new section for <code>DlgcEarlyMediaBridgeDemo</code>. <p>Appendix: JSR 309 Connector Environment Setup:</p> <ul style="list-style-type: none"> Added new section for instructions on how to set up the JSR 309 Connector environment. |

| Revision | Release Date | Notes |
|---------------------------|---------------|---|
| 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: April 2013 | | |

Refer to www.dialogic.com for product updates and for information about support policies, warranty information, and service offerings.

1. Welcome

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

Assumptions

This Quick Start 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.
- PowerMedia XMS documentation at www.dialogic.com/manuals.
- Dialogic technical support at www.dialogic.com/support.
- JSR 309 documentation on the Java Community Process website at 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 a Media Server resource.

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.

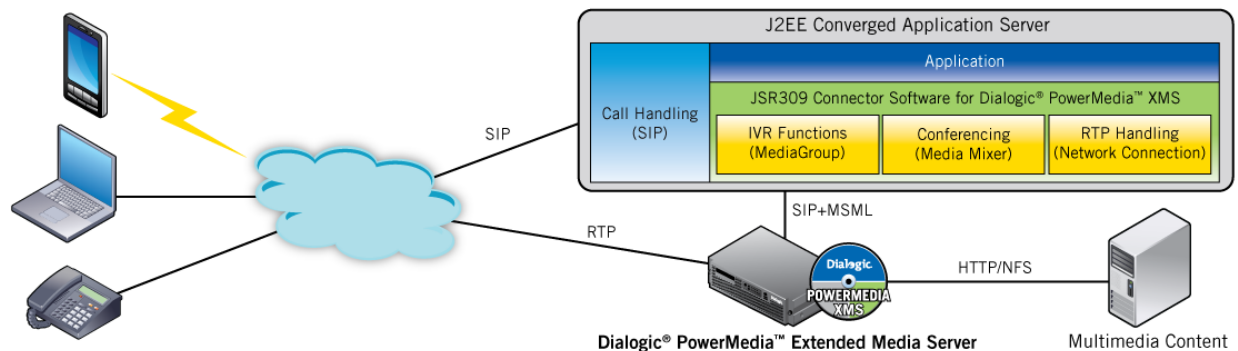
For a list of JSR 309 API parameters supported by the JSR 309 Connector, see [Supported Configurations](#).

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.

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.

Application Programming Model

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.

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

For a list of known issues, see the *Dialogic® PowerMedia™ XMS Release Notes*.

Limitations

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

Conference Limitations

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

RTC (Runtime Controls)

Various JSR 309 APIs accept 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.

Signal Detection

Pattern matching mechanism is not supported. The flushBuffer API is not implemented.

CodecPolicy

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

VxmlDialog

VxmlDialog is not supported.

Video

Video streams, video layout, and video rendering components are 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, RECEIVE, and DUPLEX. However, not all combinations are supported.

NC refers to Network Connection. MG refers to Media Group. MX refers to Media Mixer.

SEND means the media streams can flow from joiner to joinee only. RECEIVE 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:

| | NC | MG | MX |
|---------------------|-----|-----|-----|
| NC (DUPLEX) | YES | YES | YES |
| NC (SEND) | YES | N/A | YES |
| NC (RECEIVE) | YES | NO | YES |
| MG (DUPLEX) | YES | N/A | YES |
| MG (SEND) | NO | NO | NO |
| MG (RECEIVE) | NO | N/A | NO |
| MX (DUPLEX) | YES | YES | NO |
| MX (SEND) | YES | NO | NO |
| MX (RECEIVE) | YES | NO | NO |

Note: The following join combinations are the same.

```
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 Quick Start Guide assumes that you have the following systems in place before installing JSR 309 Connector:

- A working OCCAS 5.0 system for development and testing.
- A working PowerMedia XMS 2.0 system.
- SIP phones or soft clients.

Note: Contact your Dialogic representative if you need to upgrade to a newer version of PowerMedia XMS.

Supported Platforms

The JSR 309 Connector was developed using the Java SDK version 1.6. The JSR 309 Connector has been deployed and tested on OCCAS 5.0. It uses the JRockit Java Virtual Machine (JVM).

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 following files included in the .tgz package for this release:

| JSR 309 Connector Files | Description |
|--|--|
| about-log4j/ <i>dlgmisc-log4j-sample.properties</i> | Properties file used to set up log4j. |
| applications/ <i>dlgmisc_tests.war</i> | A deployable web archive that can be used to test the supported functionality. The WAR file implements several test servlets; see Test Servlets for more information. |
| lib/ <i>dlgmisc.jar</i> | An archive which contains the JSR 309 Connector implementation for the PowerMedia XMS. |
| lib/ <i>jain-sip-sdp-1.2.91.jar</i> | Third-party software; SDP library used by the JSR 309 Connector implementation. |
| lib/ <i>log4j-1.2.15.jar</i> | Third-party software for logging. |
| lib/ <i>slf4j-api-1.7.2.jar</i> | Support for Simple Logging Facade framework. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |
| lib/ <i>slf4j-log4j12-1.7.2.jar</i> | Support for Simple Logging Facade framework implementation using log4j. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |
| lib/ <i>msmltypes.jar</i> | MSML XML Bean Java Object. Required by the <i>dlgmisc.jar</i> . |
| props/ <i>dlgmiscTemplate.properties</i> | Properties file used to set up the IP address and port number, among other things, for the JSR 309 Connector and the PowerMedia XMS. |

| JSR 309 Connector Files | Description |
|--|--|
| sample-src/ <i>dlgmsc_tests.zip</i> | 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 distribution to run test servlets consists of media files included in *mscdemoprompts.tgz*. To use the media files, extract *mscdemoprompts.tgz* which will place the media files in the appropriate subdirectories.

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 5.0. If you are **not** familiar with OCCAS or how to set it up, refer to the [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 the new PowerMedia XMS via the MSML protocol. When using this release, the `mediaserver.msType=XMS` attribute must be set to XMS. This will load the correct connector implementation.

Step 1

Extract the *.tgz* distribution package for this release which will create a `<Release Package>/lib` directory.

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

Copy the following *.jar* files from `<Release Package>/lib` directory to your `<Domain Location>/lib` directory:

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

| JAR File | Description |
|--|---|
| lib/ <i>dlgmsc.jar</i> | An archive which contains the JSR 309 Connector implementation for the PowerMedia XMS. |
| lib/ <i>msmltypes.jar</i> | MSML XML Bean Java Object. Required by the <i>dlgmsc.jar</i> . |
| lib/ <i>log4j-1.2.15.jar</i> | Third-party software for logging. |
| lib/ <i>jain-sip-sdp-1.2.91.jar</i> | Third-party software; SDP library used by the JSR 309 Connector implementation. |
| lib/ <i>slf4j-api-1.7.2.jar</i> | Support for Simple Logging Facade framework. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |

| JAR File | Description |
|--|--|
| lib/ <i>slf4j-log4j12-1.7.2.jar</i> | Support for Simple Logging Facade framework implementation using log4j. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |

Step 2

Set up the properties file (*dlgmscTemplate.properties*). In order for applications to work, you must configure IP addresses and ports for the JSR 309 Connector and the PowerMedia XMS.

Follow these steps to set up the properties file:

1. Create a user directory. For example:
/root/user1
Note: This is a directory which will be accessible from the OCCAS to JSR 309 Connector configuration files.
2. Copy the *dlgmsc-log4j-sample.properties* file from <Release Package>/about-log4j to /root/user1.
3. Make a copy of the *dlgmscTemplate.properties* file which is part of the distribution. Place this file in any directory that is accessible by the OCCAS. For example:
/root/user1/user1_dlgmsc.properties
4. Edit this properties file according to your needs. Change the IP and port values for the JSR 309 Connector and the PowerMedia XMS to match the following setup:
Connector's address information (typically same as the SipServlet container)
connector.sip.address=xxx.xxx.xxx.xxx
connector.sip.port=5060

Media Server's address information
mediaserver.msType=XMS
mediaserver.1.sip.address=xxx.xxx.xxx.xxx
mediaserver.1.sip.port=5060
5. Edit the <Domain Location>/bin/startWebLogic.sh OCCAS startup script. Look for the following line:
CLASSPATH="\${SAVE_CLASSPATH}"

Add the following four lines directly after:
export DLG_PROPERTY_FILE=/root/user1/user1_dlgmsc.properties
LOG4J_OPTIONS="-Dlog4j.configuration=file:/root/user1/dlgmsc-log4j-sample.properties"
XQUERYPATH=/root/Oracle/Middleware/modules/features/weblogic.server.modules.xquery_10.3.3.0.jar
CLASSPATH="\${SAVE_CLASSPATH}:\${ORCL_HOME}/server/modules/mscontrol.jar:\${ORCL_HOME}/server/lib/jsr309-descriptor-binding.jar:\${XQUERYPATH}"
6. 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
fi
```

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

7. Save the properties file then restart the WebLogic Server.

Configuring Logging

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

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.

Installing the Demo Prompts

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

1. Copy the *mscdemoprompts.tgz* file to the PowerMedia XMS machine.
2. Unzip the file to `/tmp` directory.
3. Locate the `./snow` directory after the unzip.
4. Move `/tmp/snow` to `/var/lib/xms/media/en_US/verification` to install the demo prompts.

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

SIP URIs are required to set up the SIP phone in order for the servlets to work. 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. user name) to **player**. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created contact. You will hear the following announcement once: "Please contact your service provider". After the completion of the announcement, the application hangs up the call.

To test the application, dial the following:

```
player@<OCCAS5-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. user name) to ***dtmfPromptCollect***. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

In the SIP phone, select your newly created driver test contact. You can barge-in by pressing four DTMF digits. The application collects and prints the four digits to the *dlgmsc.log* file and immediately hangs up.

The DTMF collection is matching 770. Entering this combination, the prompt will stop playing and the application completes by hanging up the phone. Otherwise, any other combination of digits will not stop the application until the timeout is reached.

To test the application, dial the following:

```
dtmfPromptCollect@<OCCAS5-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. user name) to ***recorder***. 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.

To test the application, dial the following:

```
recorder@<OCCAS5-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. user name) to ***asyncDtmf***. 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:

```
asyncDtmf@<OCCAS5-IP-ADDRESS>
```

Conference Demos

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

| Demo Name | Functionality | Requires |
|-----------------------------|---|---|
| JMCConferenceServlet | Uses mixer conference control leg. See possible functionality see menu. | Media files needs to be installed in the PowerMedia XMS for menu to work. |
| DialogicQuickConferenceDemo | Does not use a mixer conference control leg and legs directly enter into conference – no initial IVR. | Media files needs to be installed in the PowerMedia XMS for menu to work. |
| 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. |
| DlgcReferenceConferenceDemo | Creates a conference using a non-control leg setup. | Media files needs to be installed in the PowerMedia XMS for menu to work. |

JMCConferenceServlet

This test servlet illustrates a conference 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. user name) to **DialogicConferenceDemo**. Make sure that the Web Application Server is running the *dlgmsc_tests.war* application.

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

This conference performs the following:

1. Establishes a network connection and joins it with a media group.

2. Plays a prompt for 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.
4. After the leg is in the conference, the user can enter DTMF *06. This unjoins the leg from the conference then plays an announcement "The size of the conference is". After the play completes, the leg is automatically joined back to the conference.
5. If the user enters DTMF *02, this unjoins the leg from the conference then plays an announcement "The call leg has been unjoined from the conference". Note in this case the leg stays unjoined from the conference until it is put back into the conference by the user entering DTMF *03.
6. After the leg is unjoined from the conference by entering DTMF *02, the user can rejoin the leg to the conference by entering DTMF *03. Before joining the leg back to the conference, an announcement "The call leg has rejoined the conference" is played. After the play completes, the leg rejoins the conference.

To test the application, dial the following:

```
DialogicConferenceDemo@<OCCAS5-IP-ADDRESS>
```

DialogicQuickConferenceDemo

This test servlet illustrates how to implement a simple 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. user name) to **DialogicQuickConferenceDemo**. 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 simple conference performs the following:

1. It basically joins the two legs into 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 DialogicQuickConferenceDemo:

- *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

To test the application, dial the following:

```
DialogicQuickConferenceDemo@<OCCAS5-IP-ADDRESS>
```

DlgcReferenceConferenceDemo

This test servlet illustrates how to implement a 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. user name) to **DlgcReferenceConferenceDemo**. 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 DlgcReferenceConferenceDemo:

- *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

The user can change the initial direction of legs by entering the following properties in the application JSR 309 Connector property file:

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

To test the application, dial the following:

```
DlgcReferenceConferenceDemo@<OCCAS5-IP-ADDRESS>
```

DialogicBridgeConference

This test servlet illustrates how to implement a simple conference that does not required 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. user name) to **DialogicBridgeDemo**. 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:

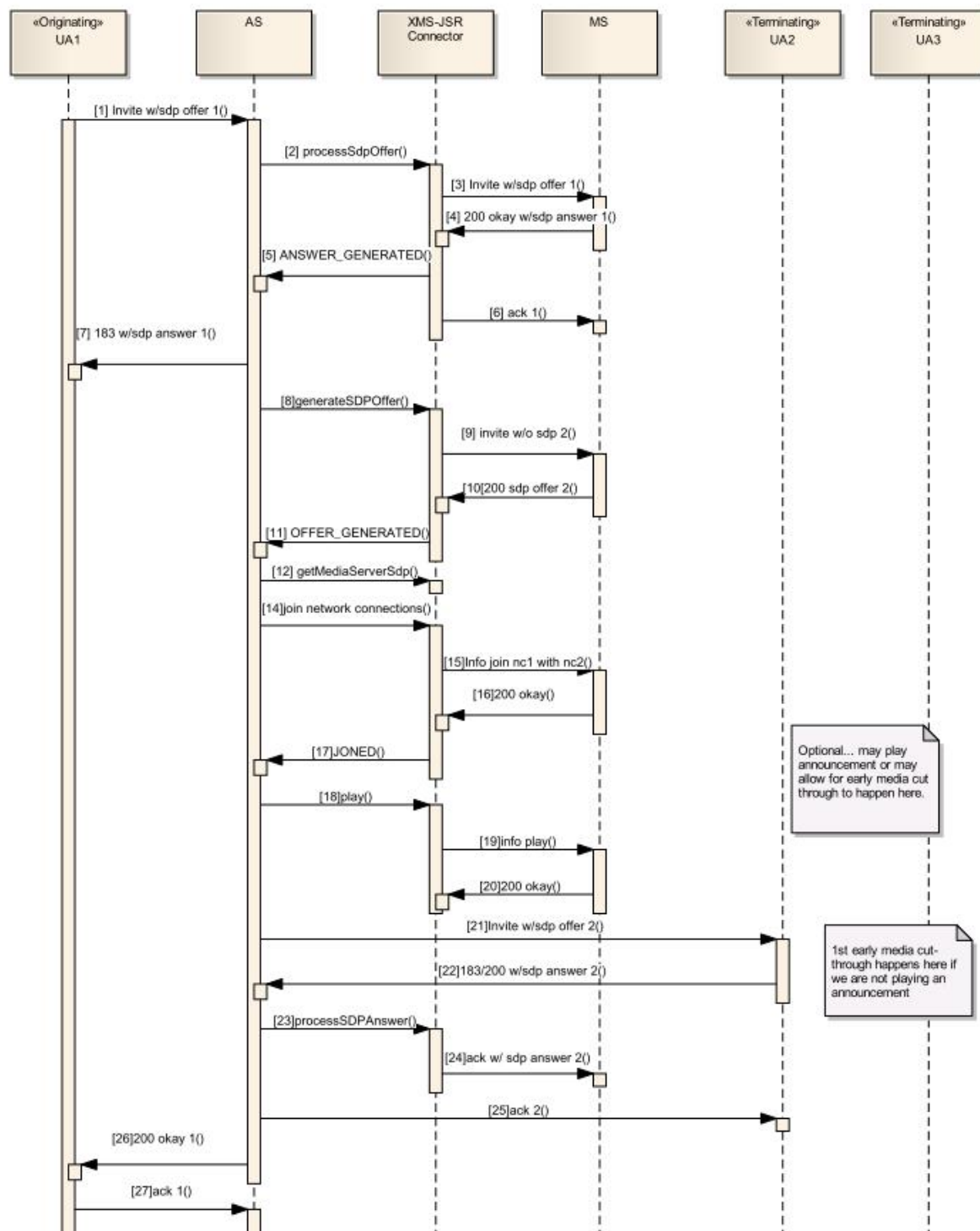
```
DialogicBridgeDemo@<OCCAS5-IP-ADDRESS>
```

DlgcEarlyMediaBridgeDemo

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

Set up your SIP phone to point to the Web Application Server. Configure the SIP phone address (i.e. user name) 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.

To test the application, dial the following:

```
DlgcEarlyMediaBridgeDemo@<OCCAS5-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 PowerMedia MSC.

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);
            }
            catch (Exception e)
            {
                log.error("doInvite: createMediaSession failed", e);
            }
        }
    }
}
```

```

        //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_dlgmisc.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 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](#).

Installing the OCCAS

This section describes the installation instructions for OCCAS 5.0. This installation illustrates how to install 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](#).

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.

Here are some highlights of the necessary steps:

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.

Make sure the database starts upon reboot:

```
chkconfig --levels 235 mysqld on
```

Specify `set-variable=lower_case_table_names=1` in `/etc/my.cnf`.

```
service mysqld restart
```


Database Setup

- Create a schema or database user for the Converged Application Server Location Service data source.
- Create a schema or database user for the User Service and Security Service data sources.

Database Creation

Note: Make a note of the databases, users, and passwords as you will need them to complete the installation process.

```
mysql> CREATE DATABASE casls;
```

```
mysql> CREATE DATABASE usss;
```

```
mysql> GRANT ALTER, CREATE, DELETE, INDEX, INSERT, LOCK TABLES, SELECT, UPDATE
ON casls.* TO 'caslsuser'@'hostname' IDENTIFIED BY 'password' WITH GRANT OPTION;
```

```
mysql> GRANT ALTER, CREATE, DELETE, INDEX, INSERT, LOCK TABLES, SELECT, UPDATE
ON usss.* TO 'usssuser'@'hostname' IDENTIFIED BY 'password' WITH GRANT OPTION;
```

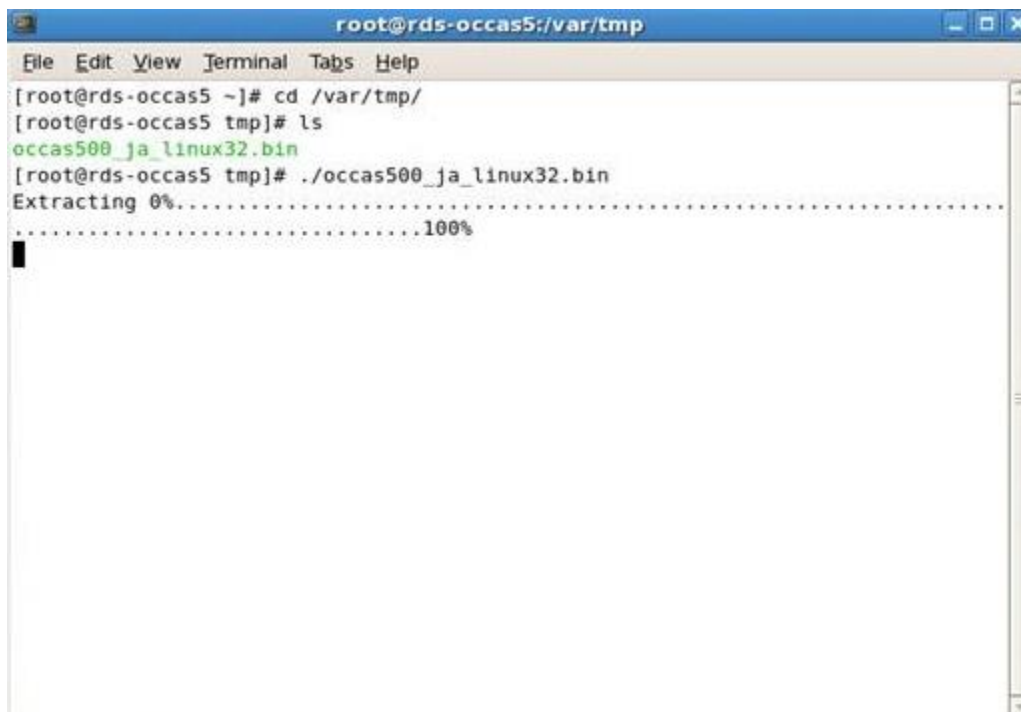
Note: hostname – should be defined in /etc/hosts file as following:

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

OCCAS .bin Installation

Run the *occas500_ja_linux32.bin* file.

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



```

root@rds-occas5:/var/tmp
File Edit View Terminal Tabs Help
[root@rds-occas5 ~]# cd /var/tmp/
[root@rds-occas5 tmp]# ls
occas500_ja_linux32.bin
[root@rds-occas5 tmp]# ./occas500_ja_linux32.bin
Extracting 0%.....100%

```



Click on **Next**.



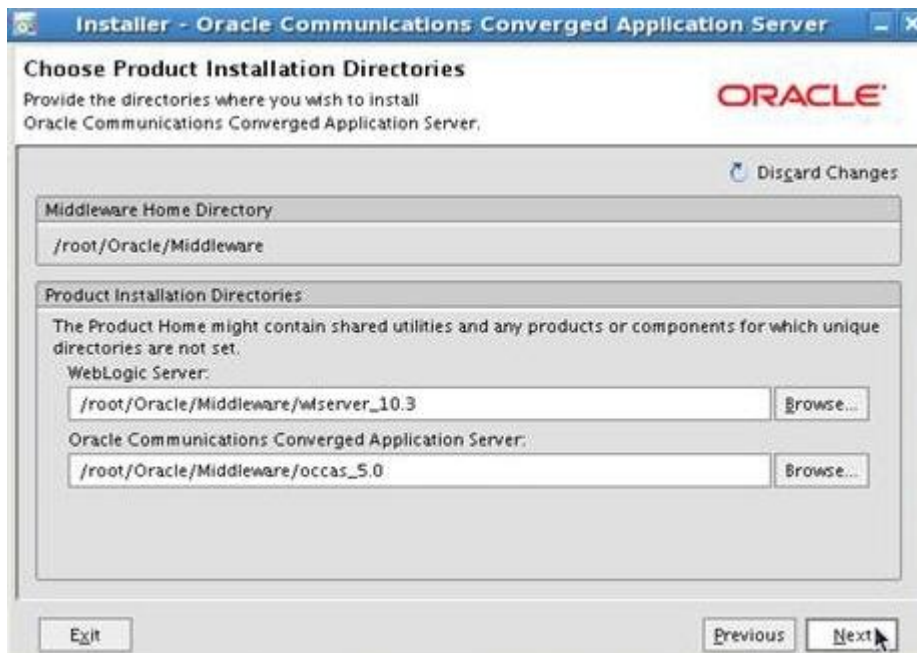
Select **Create a new Middleware Home** then click on **Next**.



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



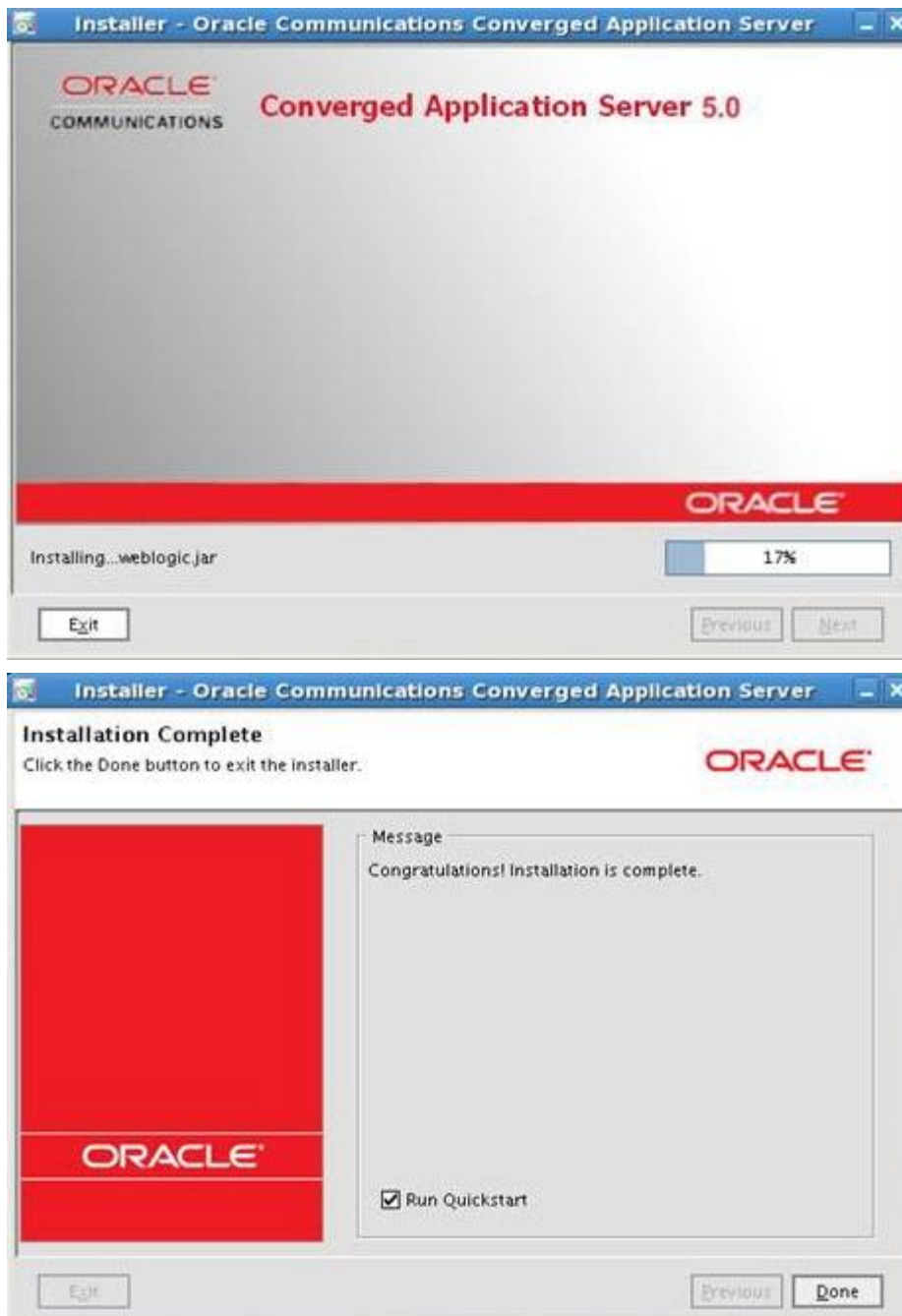
Select Typical then click on **Next**.



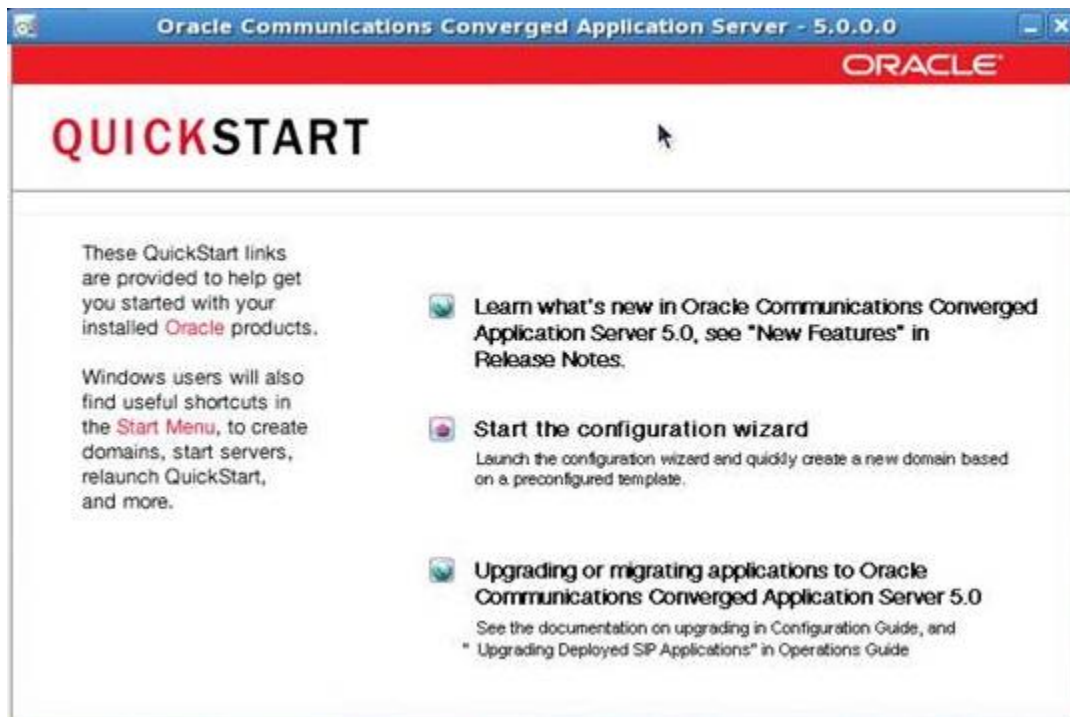
Click on **Next**.



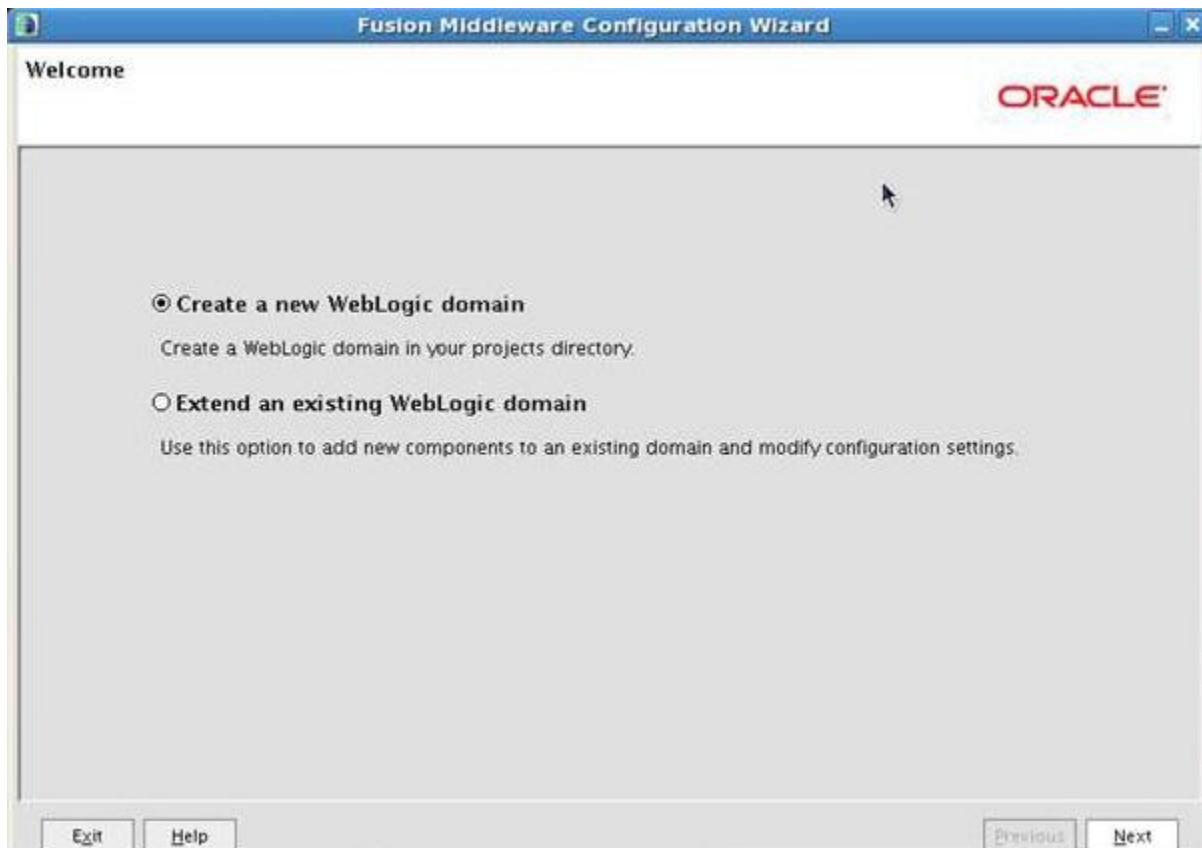
Click on **Next**.



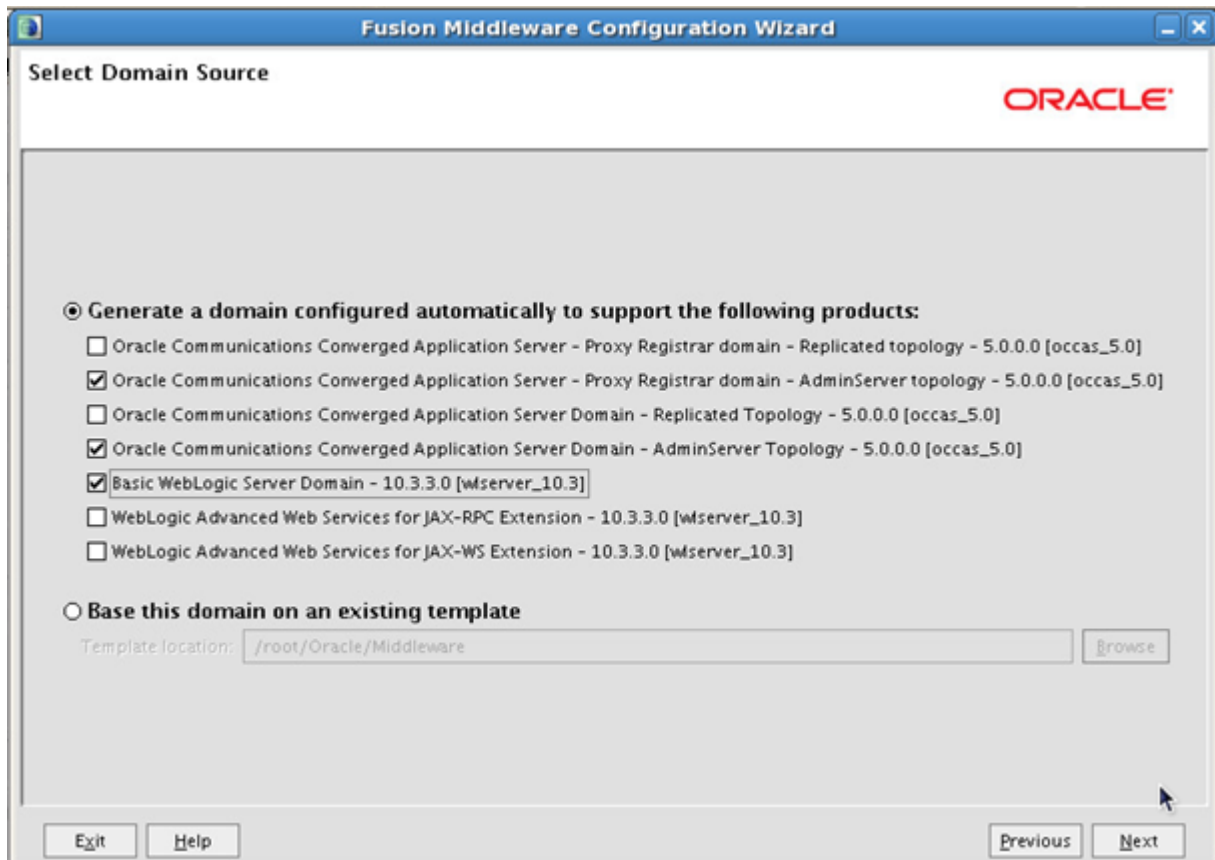
Select **Run Quickstart** then click on **Done**.



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 the following selections then click on **Next**.

- Oracle Communications Converged Application Server - Proxy Registrar domain - AdminServer topology
- Oracle Communications Converged Application Server Domain - AdminServer topology
- Basic WebLogic Server Domain



The image shows a screenshot of the 'Fusion Middleware Configuration Wizard' window. The title bar reads 'Fusion Middleware Configuration Wizard'. The main window has a header 'Specify Domain Name and Location' and the Oracle logo in the top right corner. The central area contains the instruction 'Enter the name and location for the domain:'. 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 located to the right of the 'Domain location' field. At the bottom of the window, there are four buttons: 'Exit', 'Help', 'Previous', and 'Next'.

Specify Domain Name and Location

ORACLE

Enter the name and location for the domain:

Domain name:

Domain location:

Click on **Next**.

The screenshot shows a Java Swing window titled "Fusion Middleware Configuration Wizard". The main title bar is blue with standard window controls. Below the title bar, the window has a white header area with the text "Configure Administrator User Name and Password" and the Oracle logo on the right. The main content area has a light gray background. At the top left of this area is a "Discard Changes" button with a circular arrow icon. Below this are four input fields arranged vertically. The first field is labeled "*Name:" and contains the text "weblogic". The second field is labeled "*User password:" and contains masked characters "*****". The third field is labeled "*Confirm user password:" and also contains masked characters "*****". The fourth field is labeled "Description:" and contains the text "This user is the default administrator.". At the bottom of the window, there are four buttons: "Exit", "Help", "Previous", and "Next".

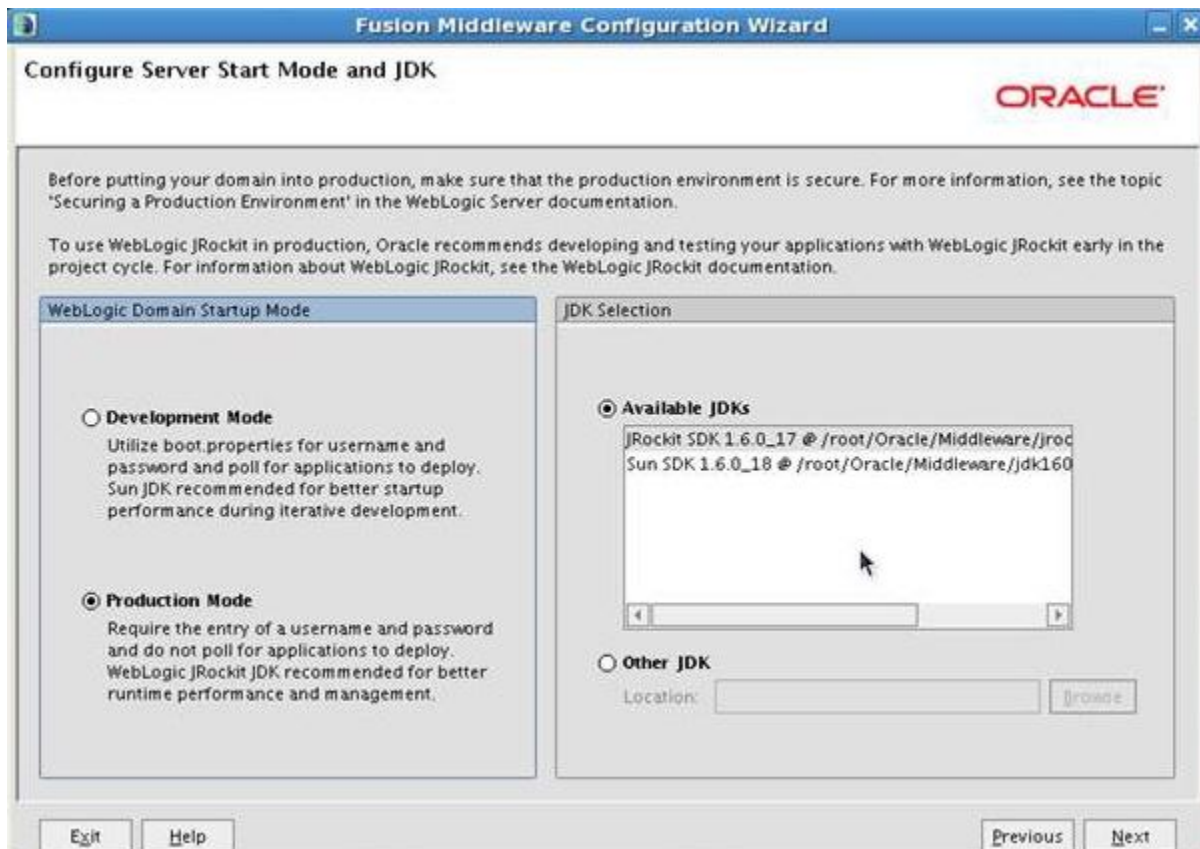
Specify **Name** and **User password** then click on Next.

The following is used as an example:

Name: weblogic

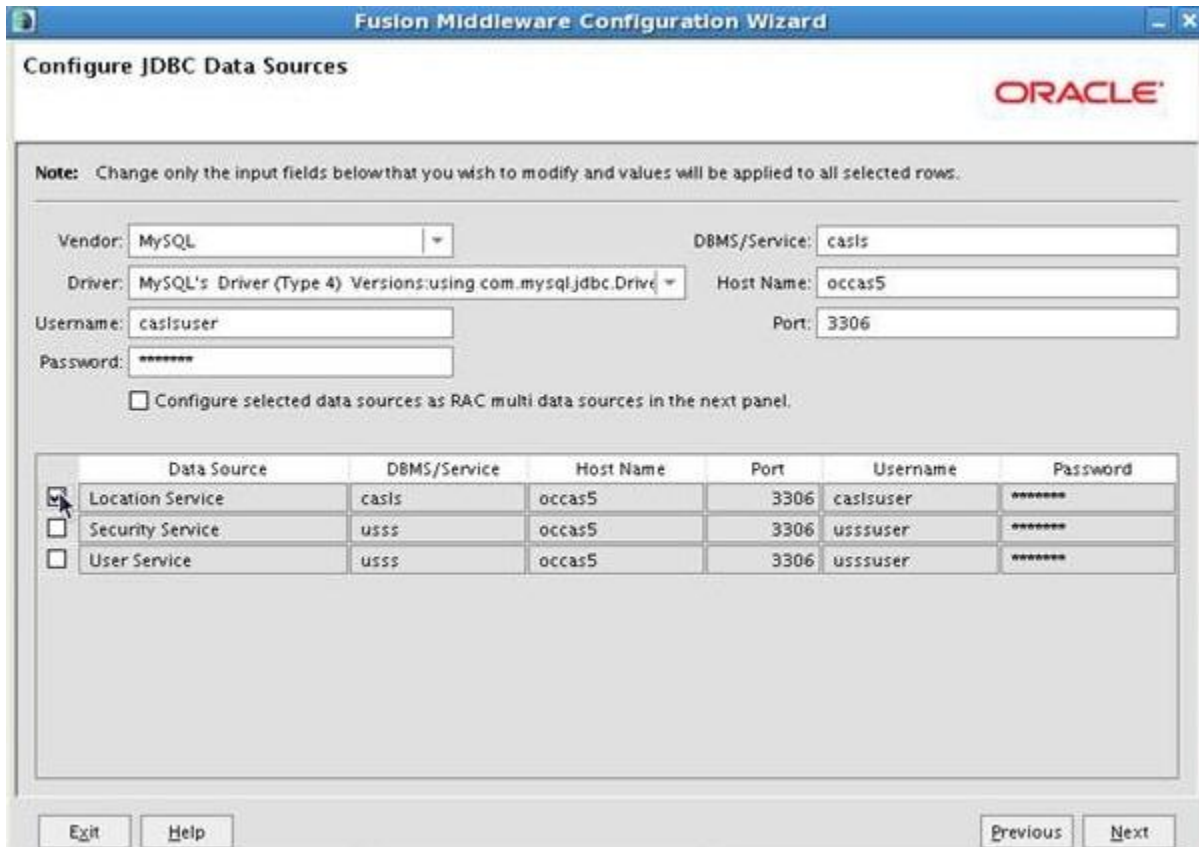
User password: Web!0gic!! ("0" is a zero)

Note: A strong password is required.



Select **Production Mode** under **WebLogic Domain Startup Mode**.

Select **Available JDKs** under **JDK Selection** and ensure that **JRockit SDK** is selected then click on **Next**.



Configure JDBC Data Sources

Note: Change only the input fields below that you wish to modify and values will be applied to all selected rows.

Vendor: MySQL DBMS/Service: casls

Driver: MySQL's Driver (Type 4) Versions: using com.mysql.jdbc.Driver Host Name: occas5

Username: caslsuser Port: 3306

Password: *****

☐ Configure selected data sources as RAC multi data sources in the next panel.

| | Data Source | DBMS/Service | Host Name | Port | Username | Password |
|-------------------------------------|------------------|--------------|-----------|------|-----------|----------|
| <input checked="" type="checkbox"/> | Location Service | casls | occas5 | 3306 | caslsuser | ***** |
| <input type="checkbox"/> | Security Service | usss | occas5 | 3306 | usssuser | ***** |
| <input type="checkbox"/> | User Service | usss | occas5 | 3306 | usssuser | ***** |

Exit Help Previous Next

Select **Location Service** then fill out the information. The information from MySQL databases, users, and passwords will be used.

Note: The information used is taken from [Database Creation](#).

Vendor: MySQL

Driver: <as listed on screen shot above>

Username: caslsuser

Password: <password>

DBMS/Service: casls

Host Name: occas5

Port: 3306

Configure JDBC Data Sources

Note: Change only the input fields below that you wish to modify and values will be applied to all selected rows.

Vendor: DBMS/Service:
 Driver: Host Name:
 Username: Port:
 Password:
☐ Configure selected data sources as RAC multi data sources in the next panel.

| | Data Source | DBMS/Service | Host Name | Port | Username | Password |
|-------------------------------------|------------------|--------------|-----------|------|-----------|----------|
| <input type="checkbox"/> | Location Service | casis | occas5 | 3306 | casisuser | ***** |
| <input checked="" type="checkbox"/> | Security Service | usss | occas5 | 3306 | ussuser | ***** |
| <input checked="" type="checkbox"/> | User Service | usss | occas5 | 3306 | ussuser | ***** |

Exit Help Previous Next

Select **Security Service** and **User Service** then fill out the information. The information from MySQL databases, users, and passwords will be used.

Note: The information used is taken from [Database Creation](#).

Vendor: MySQL

Driver: <as listed on screen shot above>

Username: usssuser

Password: <password>

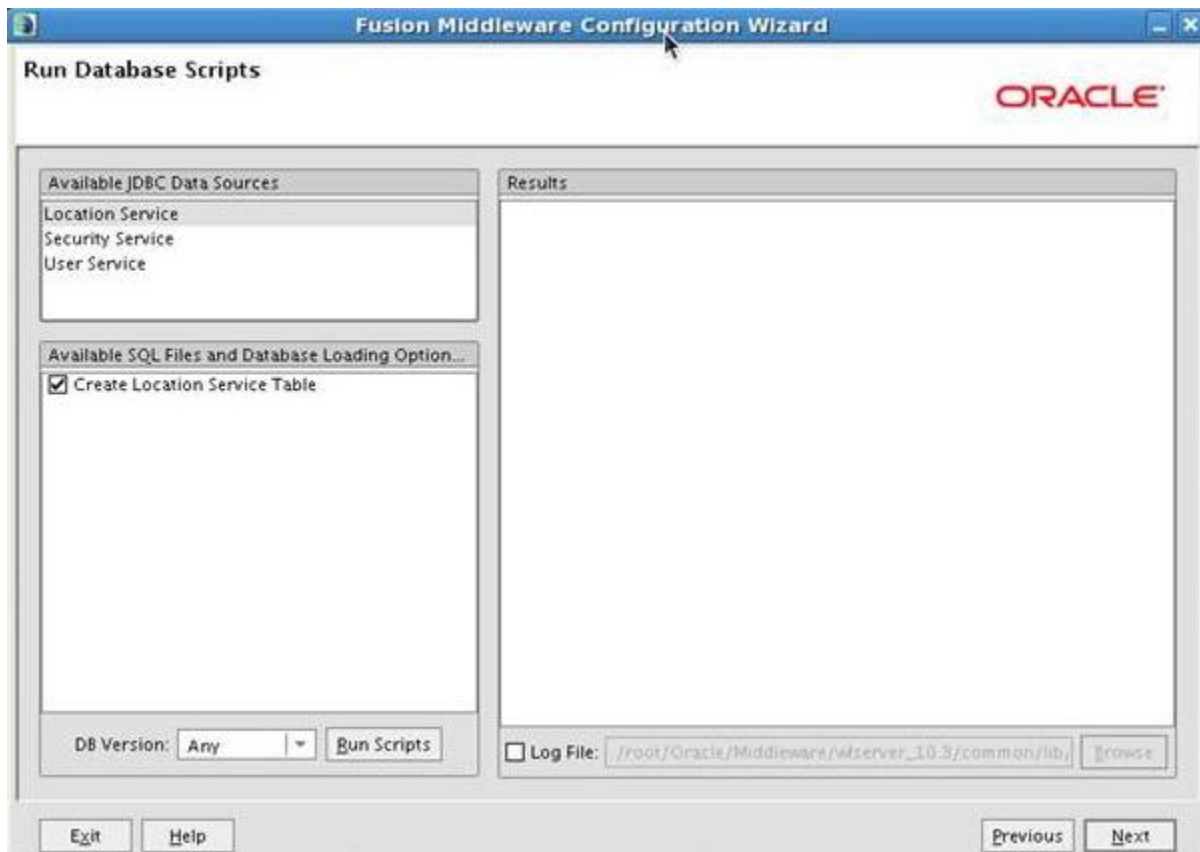
DBMS/Service: usss

Host Name: occas5

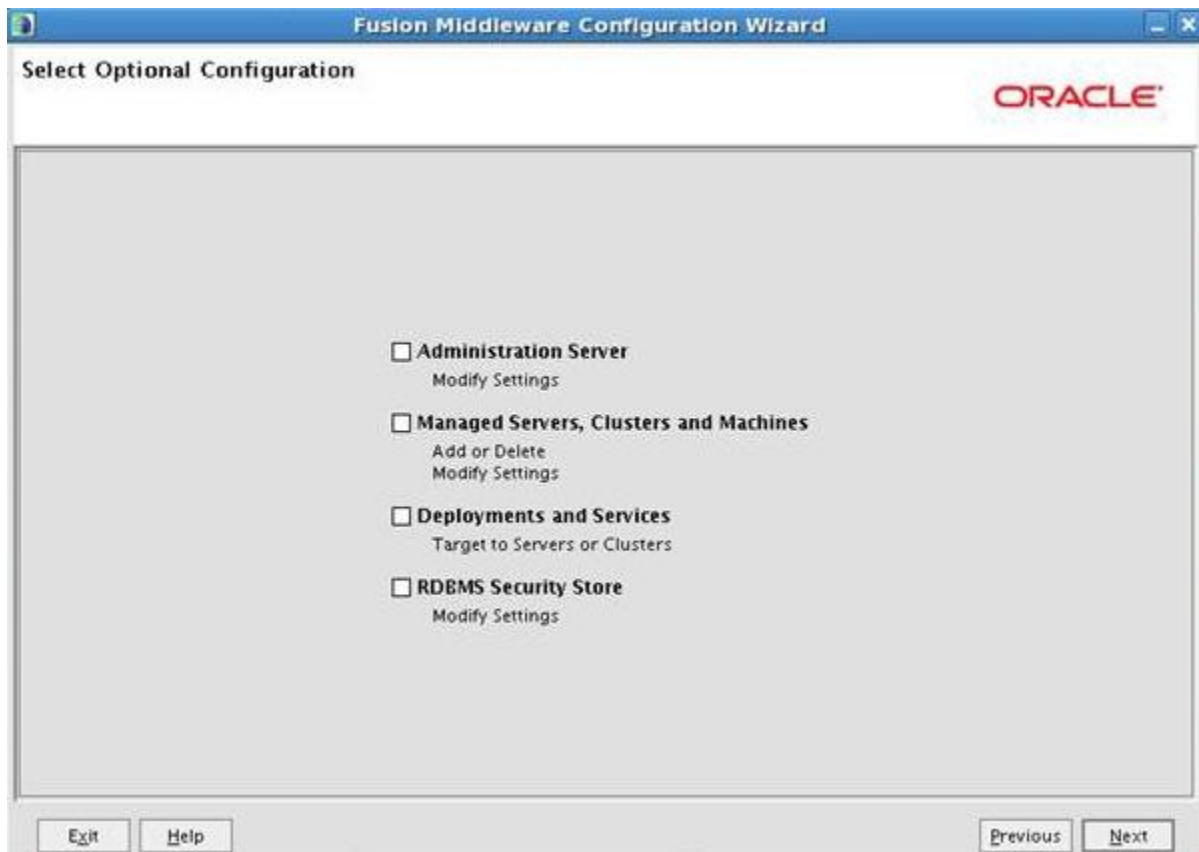
Port: 3306



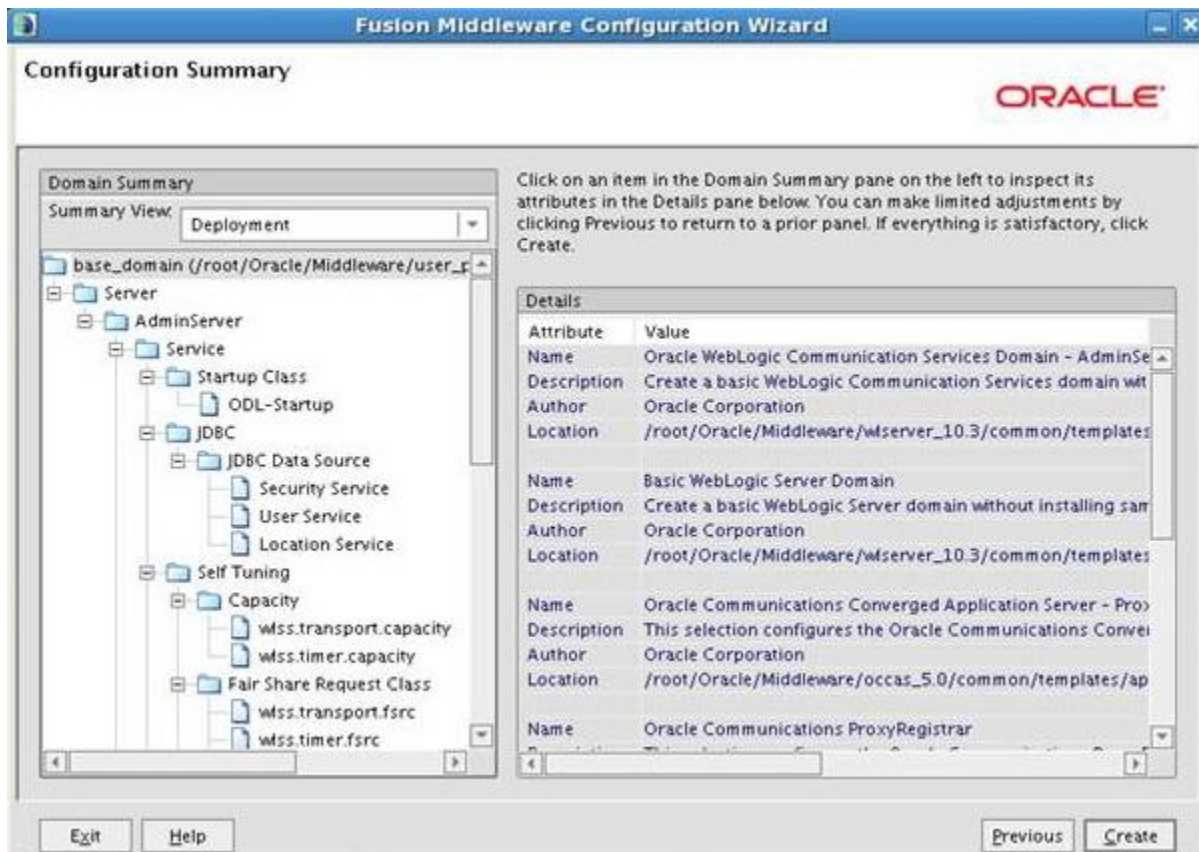
In this step, the OCCAS installation will verify if the database is set up correctly. Make sure the testing passes. If successful, click on **Next**.



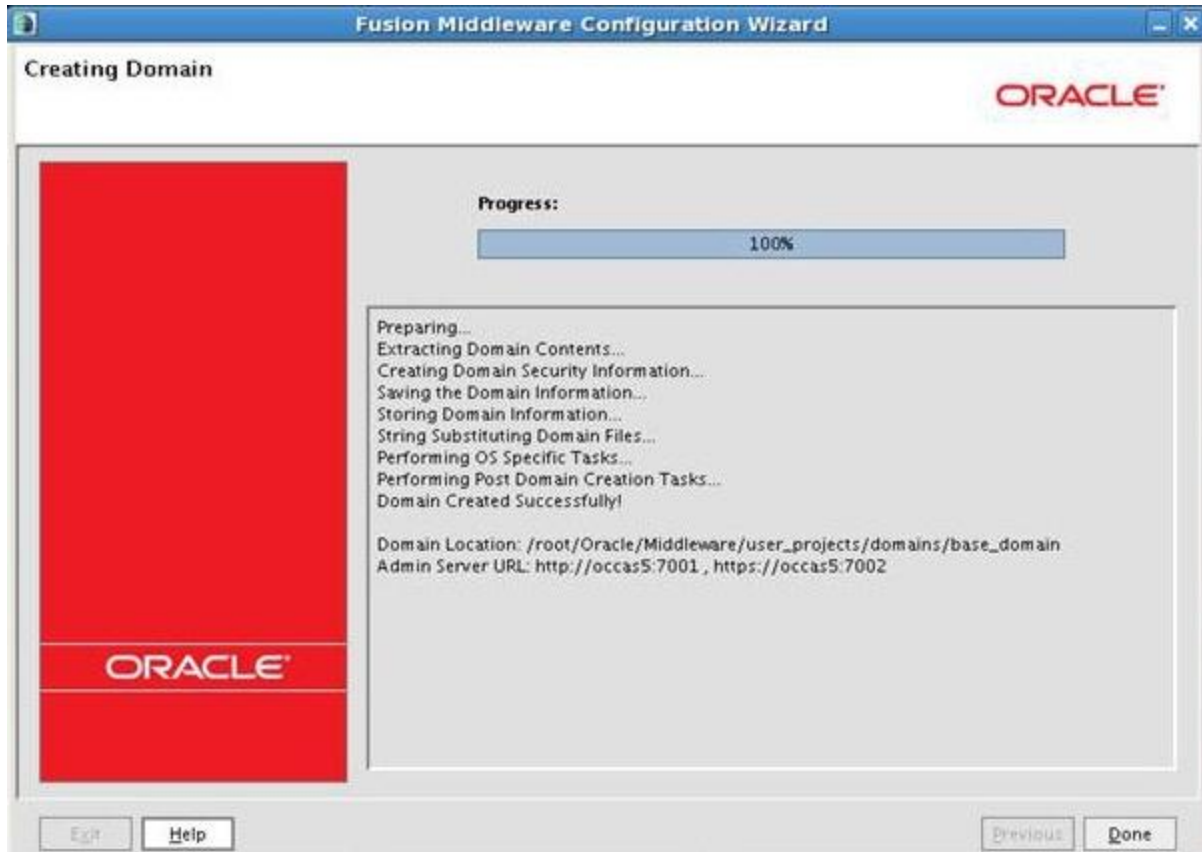
Click on **Next**.



Click on **Next**.



Click on **Create**.



The installation is complete. Click on **Done**.

OCCAS Startup

To start OCCAS, go to the **Domain Location**:

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

```

root@rds-occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
[root@rds-occas5 bin]# pwd
/root/Oracle/Middleware/user_projects/domains/base_domain/bin
[root@rds-occas5 bin]#

```

```

root@rds-occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
*****
* To start WebLogic Server, use a username and *
* password assigned to an admin-level user. For *
* server administration, use the WebLogic Server *
* console at http://hostname:port/console *
*****
starting weblogic with Java version:
Java version "1.6.0_17"
Java(TM) SE Runtime Environment (build 1.6.0_17-b04)
Oracle JRockit(R) (build R28.0.0-677-129957-1.6.0_17-20100305-1103-linux-ia32, compiled mode)
Starting WLS with line:
/root/Oracle/Middleware/jrockit_160_17_R28.0.0-677/bin/java -jrockit -Xms512m -Xmx512m -Dweblogic.Name=AdminServer -Djava.security.policy=/root/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.policy -da -Dplatform.home=/root/Oracle/Middleware/wlserver_10.3 -Dwls.home=/root/Oracle/Middleware/wlserver_10.3/server -Dweblogic.home=/root/Oracle/Middleware/wlserver_10.3/server -Ddomain.home=/root/Oracle/Middleware/user_projects/domains/base_domain -Dweblogic.management.discover=true -Dwlw.iterativeDev=false -Dwlw.testConsole=false -Dwlw.logErrorsToConsole=false -Dweblogic.ext.dirs=/root/Oracle/Middleware/patch_wls1033/profiles/default/sysext_manifest_classpath:/root/Oracle/Middleware/patch_occas500/profiles/default/sysext_manifest_classpath weblogic.Server
<Oct 26, 2012 7:20:37 AM EDT> <Notice> <WebLogicServer> <BEA-000395> <Following extensions directory contents added to the end of the classpath:
/root/Oracle/Middleware/user_projects/domains/base_domain/lib/sipactivator.jar>
<Oct 26, 2012 7:20:38 AM EDT> <Info> <Server> <BEA-002647> <The service plugin, com.oracle.core.sip.activator, was added from /root/Oracle/Middleware/user_projects/domains/base_domain/lib/sipactivator.jar.>
<Oct 26, 2012 7:20:38 AM EDT> <Info> <WebLogicServer> <BEA-000377> <Starting WebLogic Server with Oracle JRockit(R) Version R28.0.0-677-129957-1.6.0_17-20100305-1103-linux-ia32 from Oracle Corporation>
<Oct 26, 2012 7:20:40 AM EDT> <Error> <Diagnostics> <BEA-320146> <Failed to register dye PROTOCOL_SIP at index 33.>
<Oct 26, 2012 7:20:40 AM EDT> <Info> <Management> <BEA-141107> <Version: WebLogic Server 10.3.3.0 Fri Apr 9 00:05:28 PDT 2010 1321401>
<Oct 26, 2012 7:20:43 AM EDT> <Info> <Security> <BEA-090065> <Getting boot identity from user.>
Enter username to boot WebLogic server:weblogic
Enter password to boot WebLogic server:

```

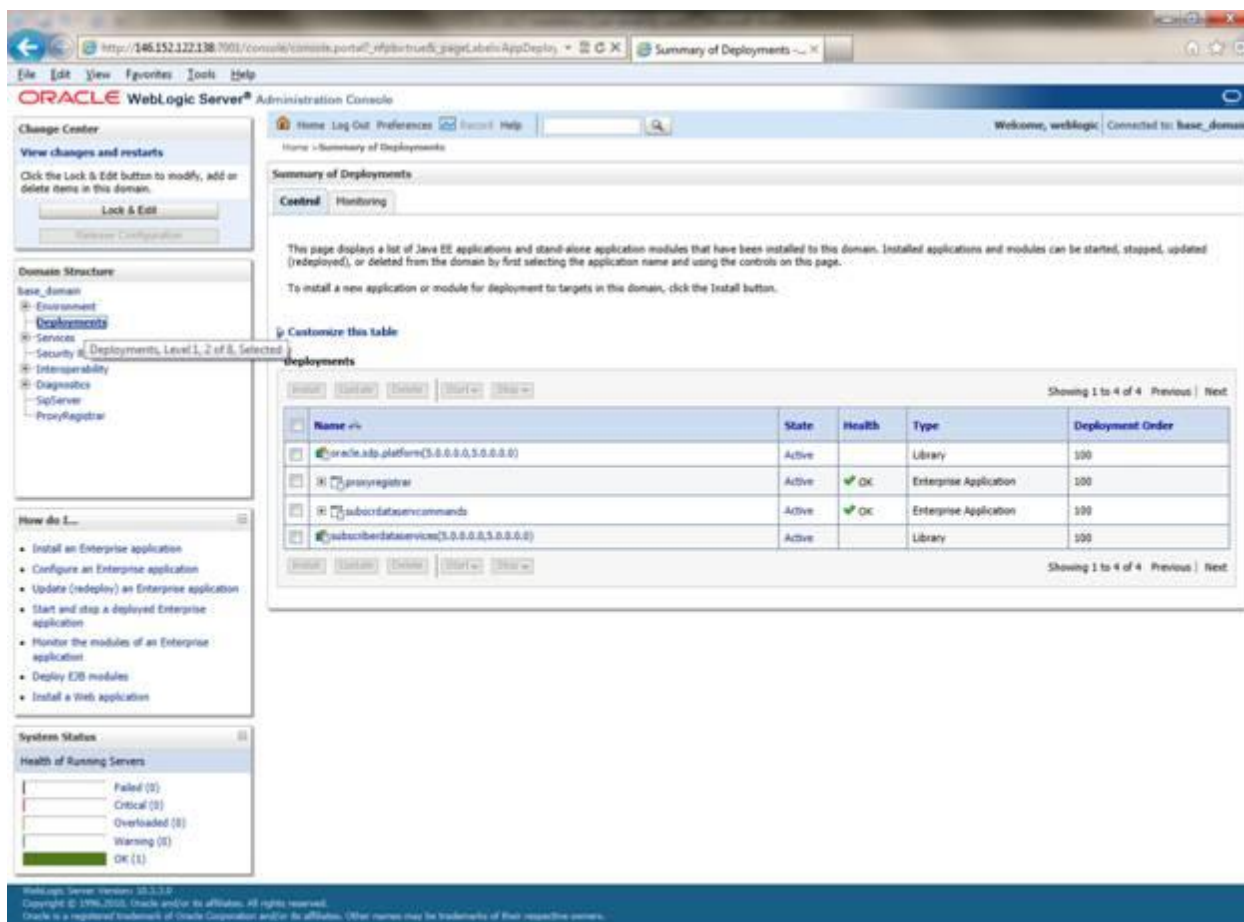
```

root@rds-occas5:~/Oracle/Middleware/user_projects/domains/base_domain/bin
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Security> <BEA-090898> <Ignoring the trusted CA certificate "CN=GlobalSign,O=GlobalSign,OU=GlobalSign Root CA - R3". The loading of the trusted certificate list raised a certificate parsing exception PKIX: Unsupported OID in the AlgorithmIdentifier object: 1.2.840.113549.1.1.11.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Security> <BEA-090898> <Ignoring the trusted CA certificate "OU=Security Communication RootCA2,O=SECOM Trust Systems CO.,LTD.,C=JP". The loading of the trusted certificate list raised a certificate parsing exception PKIX: Unsupported OID in the AlgorithmIdentifier object: 1.2.840.113549.1.1.11.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Security> <BEA-090898> <Ignoring the trusted CA certificate "CN=KEYNECTIS ROOT CA,OU=ROOT,O=KEYNECTIS,C=FR". The loading of the trusted certificate list raised a certificate parsing exception PKIX: Unsupported OID in the AlgorithmIdentifier object: 1.2.840.113549.1.1.11.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "sips" is now listening on 146.152.122.138:5061 for protocols sips.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure[1]" is now listening on 127.0.0.1:7002 for protocols iiops, t3s, ldaps, https.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "sips[1]" is now listening on 127.0.0.1:5061 for protocols sips.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "sip[1]" is now listening on 127.0.0.1:5060 for protocols sip.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "sip" is now listening on 146.152.122.138:5060 for protocols sip.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "Default[1]" is now listening on 127.0.0.1:7001 for protocols iiop, t3, ldap, snmp, http.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "Default" is now listening on 146.152.122.138:7001 for protocols iiop, t3, ldap, snmp, http.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure" is now listening on 146.152.122.138:7002 for protocols iiops, t3s, ldaps, https.>
<Oct 26, 2012 7:22:57 AM EDT> <Notice> <WebLogicServer> <BEA-000329> <Started WebLogic Admin Server "AdminServer" for domain "base_domain" running in Production Mode>
<Oct 26, 2012 7:22:58 AM EDT> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport UDP)" is listening on port 5060>
<Oct 26, 2012 7:22:58 AM EDT> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<Oct 26, 2012 7:22:58 AM EDT> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>

```

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

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



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 the new PowerMedia XMS via the MSML protocol. When using this release, the `mediaserver.msType=XMS` attribute must be set to XMS. This will load the correct connector implementation.

Step 1

Extract the *.tgz* distribution package for this release which will create a <Release Package>/lib directory.

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

Copy the following *.jar* files from <Release Package>/lib directory to your <Domain Location>/lib directory:

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

| JAR File | Description |
|--|--|
| lib/ <i>dlgmisc.jar</i> | An archive which contains the JSR 309 Connector implementation for the PowerMedia XMS. |
| lib/ <i>msmltypes.jar</i> | MSML XML Bean Java Object. Required by the <i>dlgmisc.jar</i> . |
| lib/ <i>log4j-1.2.15.jar</i> | Third-party software for logging. |
| lib/ <i>jain-sip-sdp-1.2.91.jar</i> | Third-party software; SDP library used by the JSR 309 Connector implementation. |
| lib/ <i>slf4j-api-1.7.2.jar</i> | Support for Simple Logging Facade framework. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |
| lib/ <i>slf4j-log4j12-1.7.2.jar</i> | Support for Simple Logging Facade framework implementation using log4j. Note: Refer to www.slf4j.org/manual.html for more information on different available logging implementations. |

Step 2

Set up the properties file (*dlgmiscTemplate.properties*). In order for applications to work, you must configure IP addresses and ports for the JSR 309 Connector and the PowerMedia XMS.

Follow these steps to set up the properties file:

1. Create a user directory. For example:
/root/user1
Note: This is a directory which will be accessible from the OCCAS to JSR 309 Connector configuration files.
2. Copy the *dlgmisc-log4j-sample.properties* file from <Release Package>/about-log4j to /root/user1.
3. Make a copy of the *dlgmiscTemplate.properties* file which is part of the distribution. Place this file in any directory that is accessible by the OCCAS. For example:
/root/user1/user1_dlgmisc.properties

4. Edit this properties file according to your needs. Change the IP and port values for the JSR 309 Connector and the PowerMedia XMS to match the following setup:
Connector's address information (typically same as the SipServlet container)

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

Media Server's address information

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

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

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

Add the following four lines directly after:

```
export DLG_PROPERTY_FILE=/root/user1/user1_dlgmsc.properties
LOG4J_OPTIONS="-Dlog4j.configuration=file:/root/user1/dlgmsc-log4j-
sample.properties"
XQUERYPATH=/root/Oracle/Middleware/modules/features/weblogic.server.mod
ules.xquery_10.3.3.0.jar
CLASSPATH="${SAVE_CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:
${ORCL_HOME}/server/lib/jsr309-descriptor-binding.jar:${XQUERYPATH}"
```

6. 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
fi
```

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

7. Save the properties file then restart the WebLogic Server.

Installing and Configuring the Test Servlets

Copy *dlgmisc_tests.war* from <Release Package>/applications to /root/user1 (or any other directory which will be accessible by OCCAS).

The screenshot displays the Oracle WebLogic Server Administration Console. The top menu bar includes File, Edit, View, Favorites, Tools, and Help. The main title is "ORACLE WebLogic Server® Administration Console".

On the left, the "Change Center" section has a "View changes and restarts" link and instructions: "Click the Lock & Edit button to modify, add or delete items in this domain." Below this are "Lock & Edit" and "Release Configuration" buttons.

The "Domain Structure" tree on the left shows the following hierarchy:

- base_domain
 - Environment
 - Deployments**
 - Services
 - Security Realms
 - Interoperability
 - Diagnostics
 - SipServer
 - ProxyRegistrar

The "Summary of Deployments" page is active, showing a "Control" tab and a "Monitoring" tab. The page text states: "This page displays a list of Java EE applications and (redeployed), or deleted from the domain by first s". Below this, it says: "To install a new application or module for deployment".

A "Customize this table" link is present. The "Deployments" table has buttons for "Install", "Update", "Delete", "Start", and "Stop". The table header is "Name" with an upward arrow. The first entry is "oracle.sdp.platform(5.0.0.0.0,5.0.0.0.0)".

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

File Edit View Favorites Tools Help

ORACLE WebLogic Server® Administration Console

Home Log Out Preferences Record Help

Home > Summary of Deployments

Summary of Deployments

Control Monitoring

This page displays a list of Java EE applications and stand-alone applications (redeployed), or deleted from the domain by first selecting the application.

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

[Customize this table](#)

Deployments

| | Name | Install | Update | Delete | Start | Stop |
|--|---|---------|--------|--------|-------|------|
| | oracle.sdp.platform(5.0.0.0.5.0.0.0.0) | | | | | |
| | proxyregistrar | | | | | |
| | subscrdataservcommands | | | | | |
| | subscriberdataservices(5.0.0.0.0,5.0.0.0.0) | | | | | |

Sort table by Name

Change Center

View changes and restarts

No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

Lock & Edit

Release Configuration

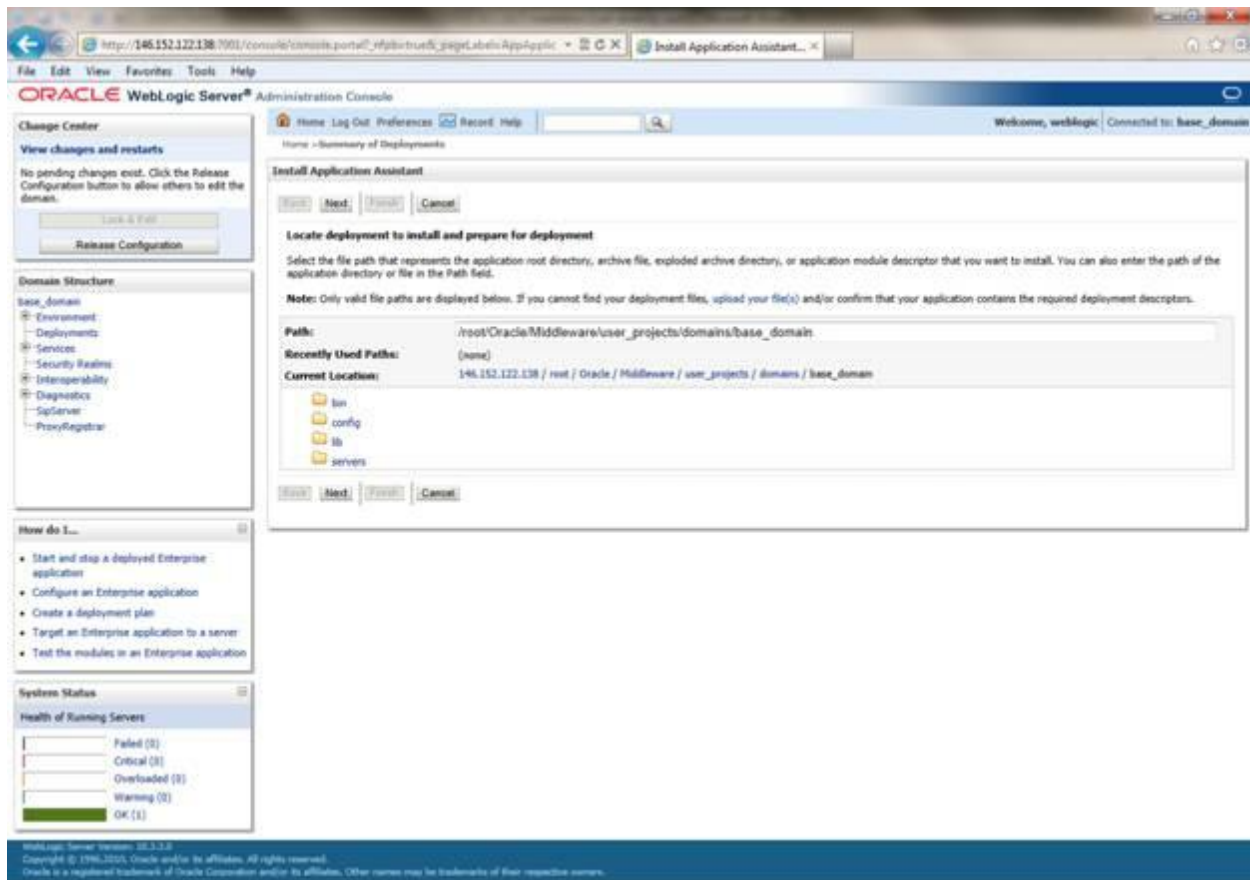
Domain Structure

- base_domain
 - Environment
 - Deployments**
 - Services
 - Security Realms
 - Interoperability
 - Diagnostics
 - SipServer
 - ProxyRegistrar

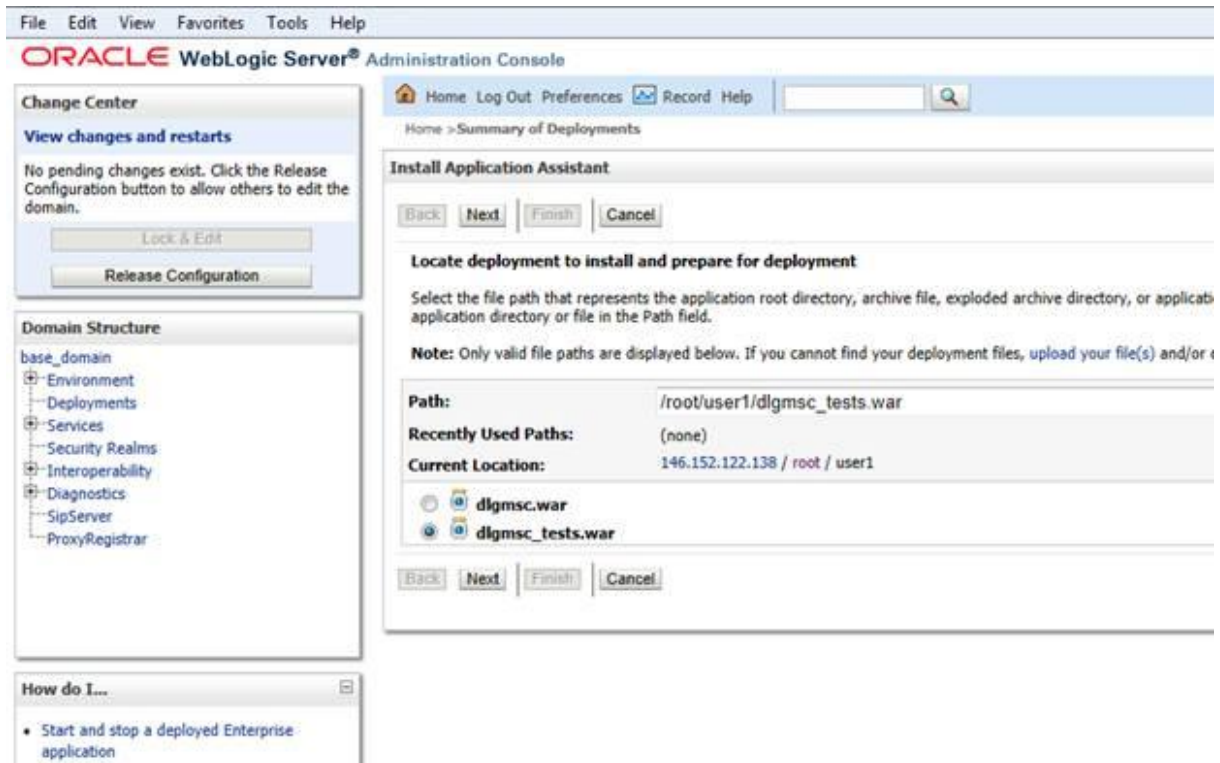
How do I...

- Install an Enterprise application
- Configure an Enterprise application
- Update (redeploy) an Enterprise application
- Start and stop a deployed Enterprise application
- Monitor the modules of an Enterprise application

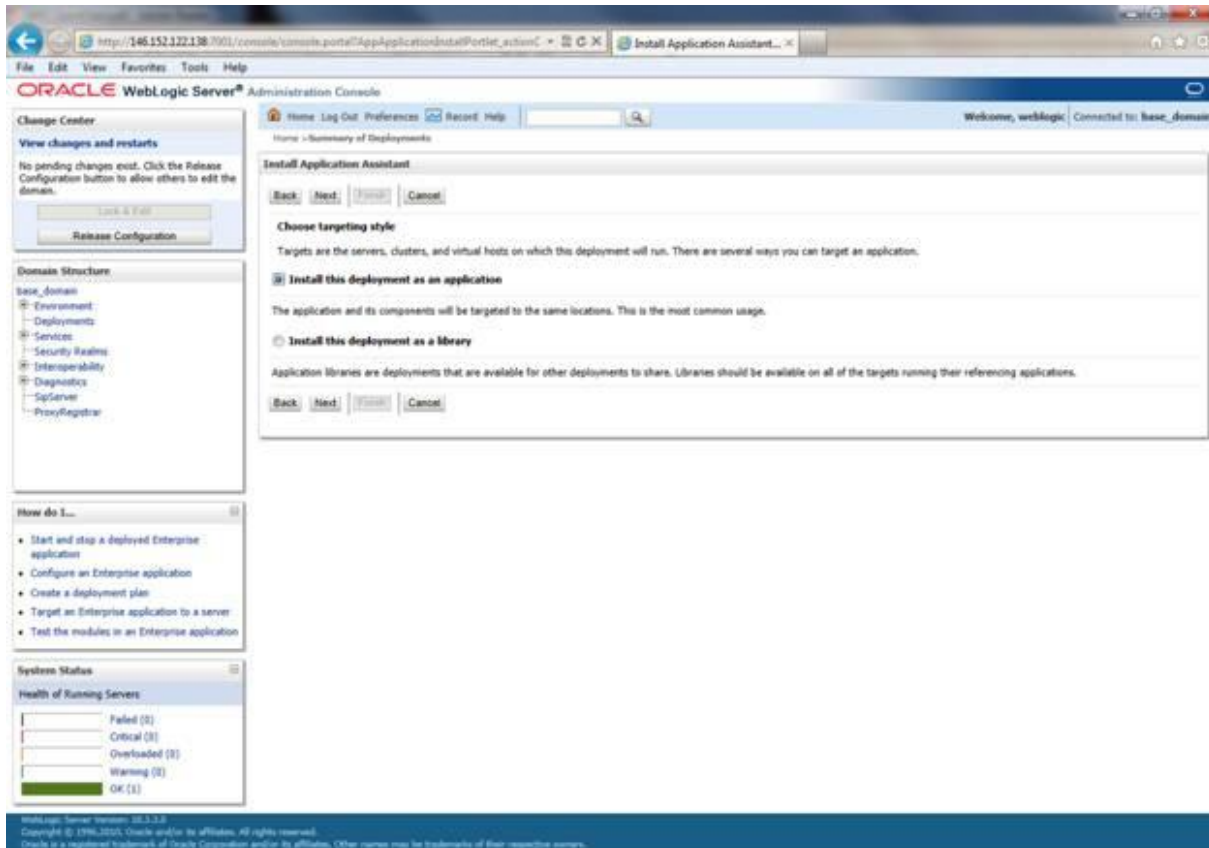
Click on **Install** under **Deployments**.



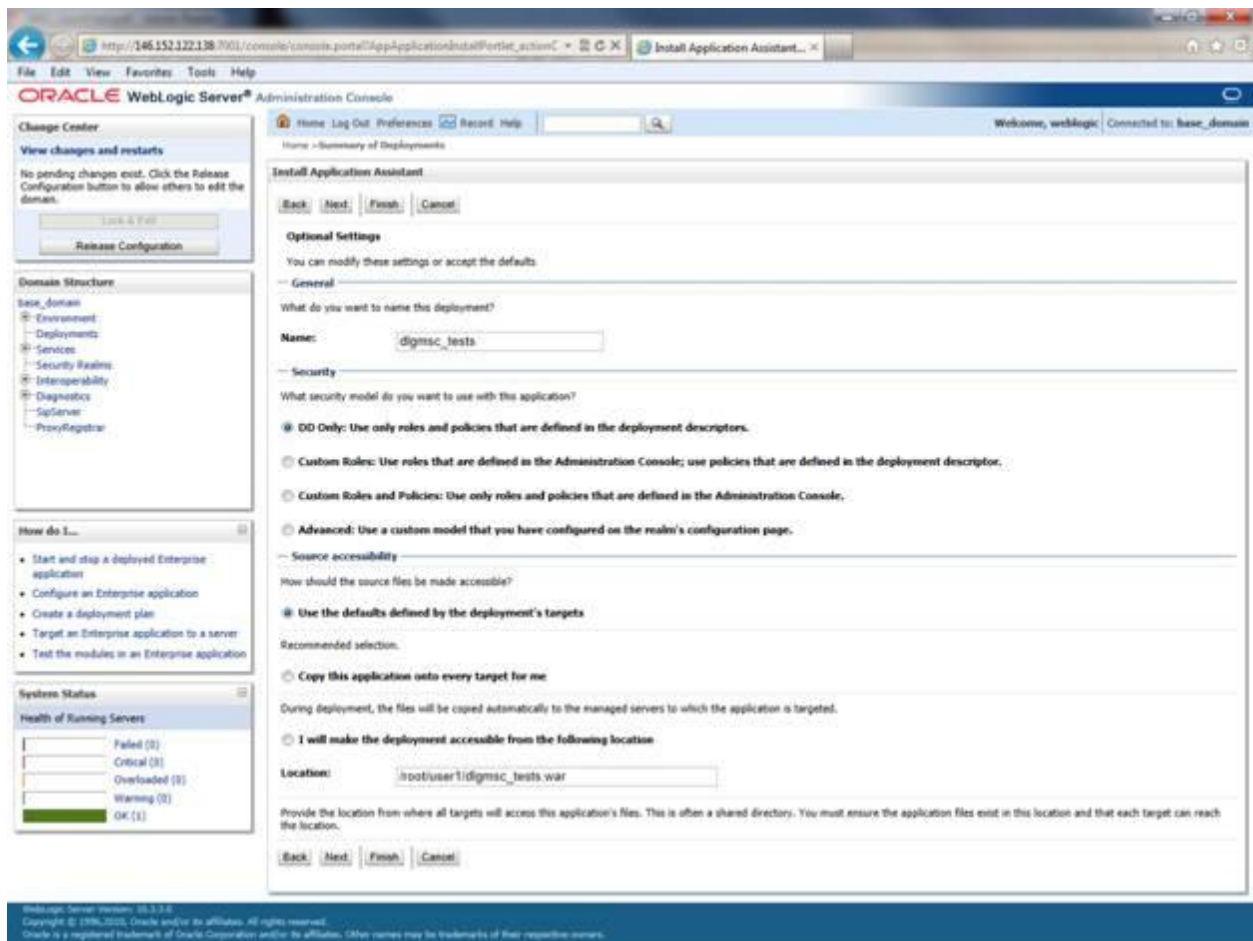
Change the **Path** to `/root/user1` by clicking on **root** under **Current Location** and then **user1**.



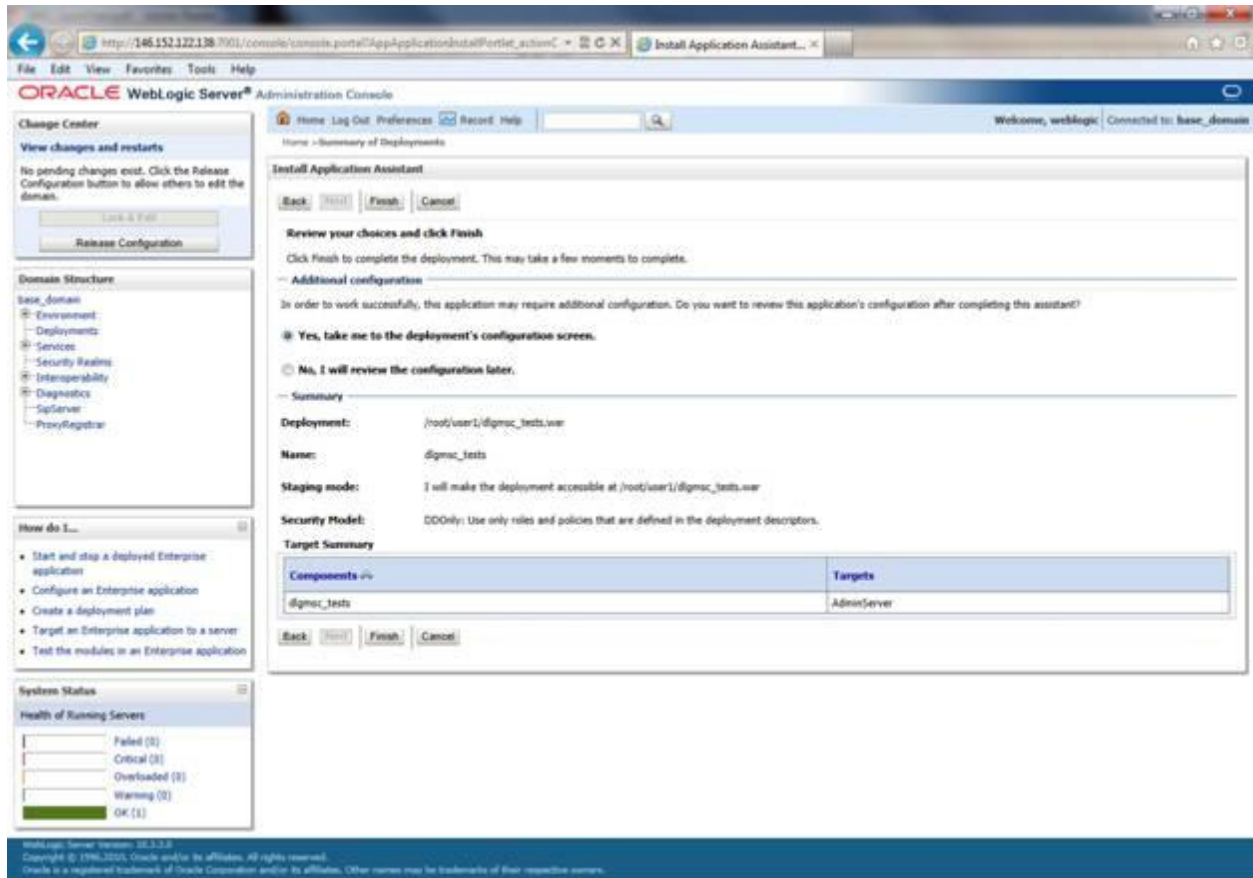
Select **dlgmsc_tests.war** then click on **Next**.



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



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



Click on **Finish**.

Oracle WebLogic Server® Administration Console

Home Log Out Preferences Record Help

Welcome, weblogic Connected to: base_domain

Home » Summary of Deployments » dlgmisc_tests

Settings for dlgmisc_tests

Overview Deployment Plan Configuration Security Targets Control Testing Monitoring Notes

Save

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

Name: dlgmisc_tests The name of this application deployment. [More Info...](#)

Context Root: dlgmisc_tests The specific path at which this web application is found by a servlet. [More Info...](#)

Path: / root/ user/ dlgmisc_tests. war The path to the source of the deployable unit on the Administration Server. [More Info...](#)

Deployment Plan: (no plan specified) The path to the deployment plan document on Administration Server. [More Info...](#)

Staging Mode: nostage The mode that specifies whether an application's files are copied from a source on the Administration Server to the Managed Server's staging area during application preparation. [More Info...](#)

Security Model: DDOnly The security model specifies how this deployment should be secured. [More Info...](#)

Deployment Order: 100 An integer value that indicates when this unit is deployed, relative to other deployable units on a server, during startup. [More Info...](#)

Deployment Principal Name: A string value that indicates what principal should be used when deploying the file or archive during startup and shutdown. This principal will be used to set the current subject when calling out into application code for interfaces such as ApplicationsLifecycleListener. If no principal name is specified, then the anonymous principal will be used. [More Info...](#)

Save

Modules and Components

Showing 1 to 1 of 1 Previous Next

| Name | Type |
|-----------------|-----------------|
| dlgmisc_tests | Web Application |
| Web Services | |
| None to display | |

Showing 1 to 1 of 1 Previous Next

WebLogic Server Version: 10.3.6.0
Copyright © 1996, 2006. 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.

Click on **Save** then **Activate Changes**.

Oracle WebLogic Server® Administration Console

Home Log Out Preferences Record Help

Welcome, weblogic Connected to: base_domain

Home » Summary of Deployments » dlgmisc_tests » Summary of Deployments

Summary of Deployments

Control Monitoring

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from 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 button.

Customize this table

Deployments

Showing 1 to 5 of 5 Previous Next

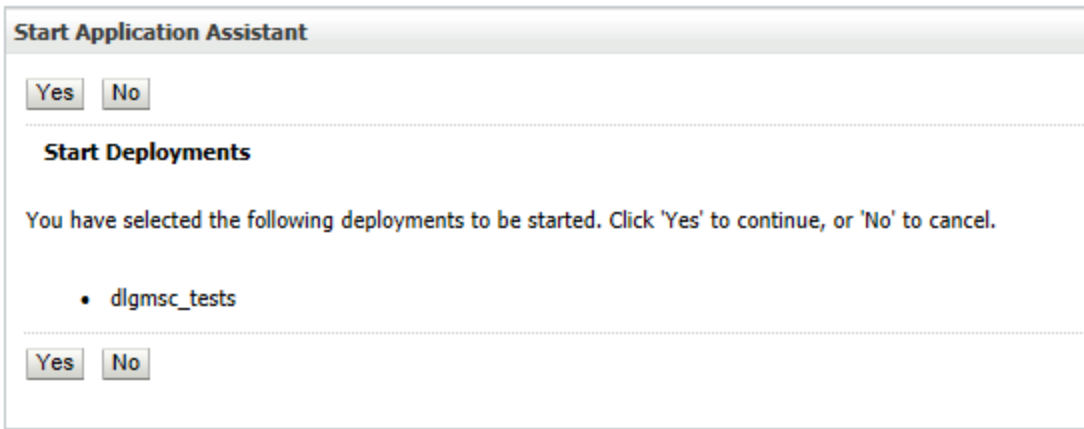
| Name | State | Health | Type | Deployment Order |
|--|----------|--------|------------------------|------------------|
| dlgmisc_tests | Prepared | OK | Web Application | 100 |
| oracle.sdp.platform(3.0.0.0.5.0.0.0.0) | Active | OK | Library | 100 |
| proxyregistrator | Active | OK | Enterprise Application | 100 |
| subscribeservicecommands | Active | OK | Enterprise Application | 100 |
| subscribeservices(3.0.0.0.5.0.0.0.0) | Active | OK | Library | 100 |

Showing 1 to 5 of 5 Previous Next

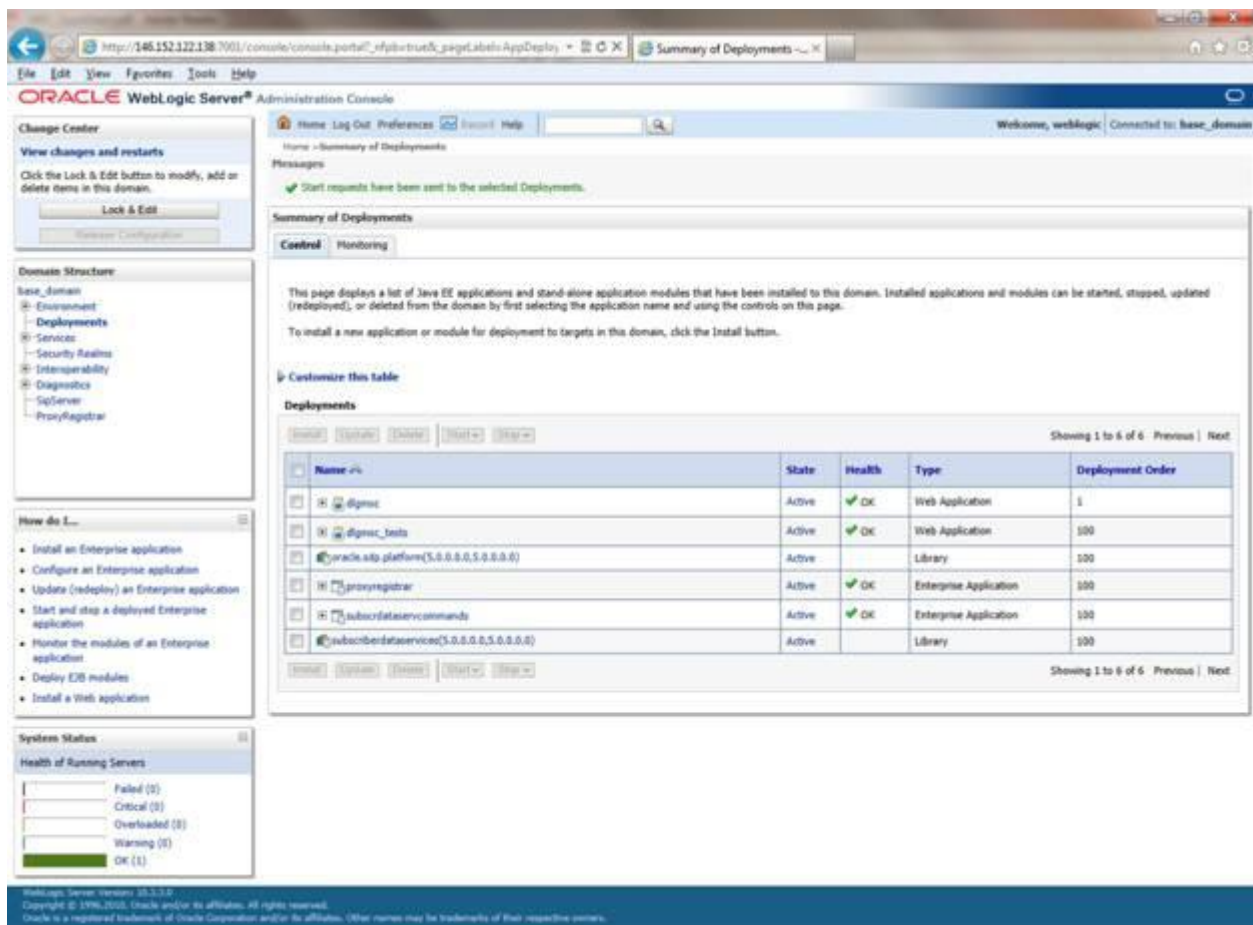
WebLogic Server Version: 10.3.6.0
Copyright © 1996, 2006. Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Go back to **Deployments** under **Domain Structure** to verify that *dlgmisc_tests.war* is in a Prepared State.

Click on **Start** then **Servicing All Requests**.



Click on **Yes**.



Verify that deployed *dlgmsc_tests.war* application is in Active State.

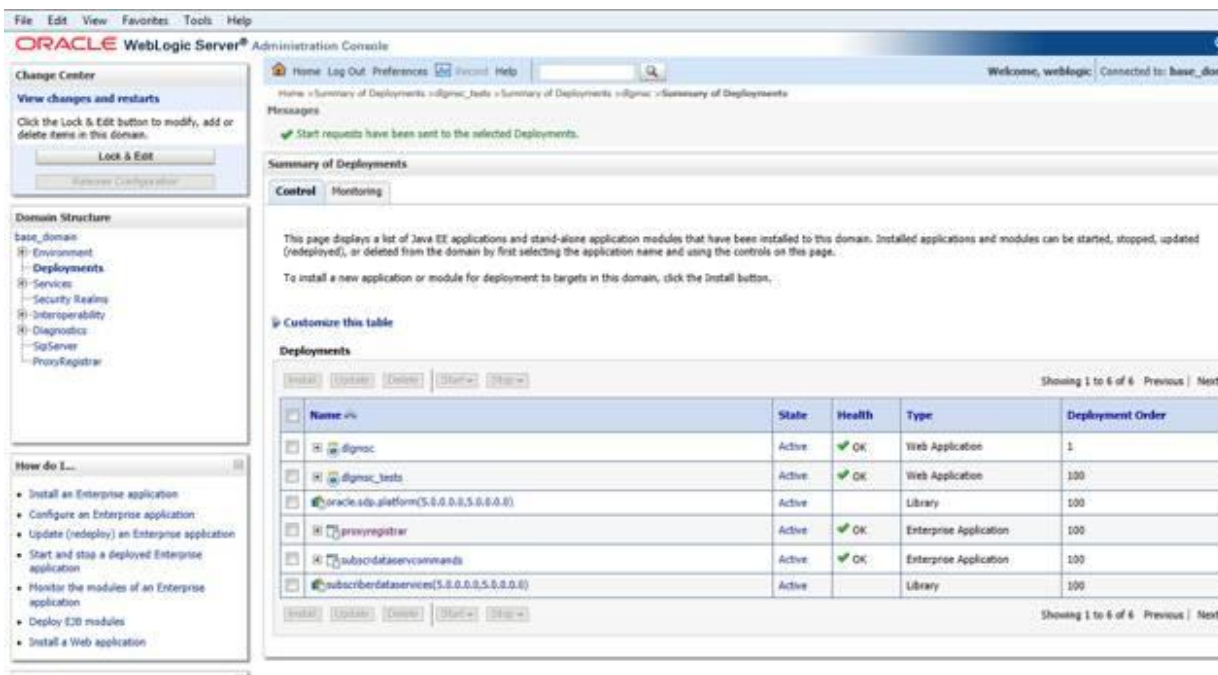
Running the Test Servlets

Test servlets are provided to illustrate the use of the JSR 309 Connector. Since these test servlets will be using media files, you have to upload files to the PowerMedia XMS which will be used in this setup.

To use the media files, extract *mscdemoprompts.tgz* in the PowerMedia XMS under `/var/lib/xms/media/en-US/verification` directory. This will extract the media files in the appropriate subdirectories.

For illustration purposes, let's remove the **proxyregistrar** deployment for a proof of concept setup.

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



The screenshot shows the Oracle WebLogic Server Administration Console. On the left, the 'Domain Structure' tree is expanded to 'Deployments'. The 'Lock & Edit' button is visible. The main area shows a 'Summary of Deployments' table with the following data:

| Name | State | Health | Type | Deployment Order |
|--|--------|--------|------------------------|------------------|
| diagnostic | Active | OK | Web Application | 1 |
| diagnostic_tests | Active | OK | Web Application | 100 |
| oracle.sdp.platform(5.0.0.0.5.0.0.0.0) | Active | | Library | 100 |
| proxyregistrar | Active | OK | Enterprise Application | 100 |
| subscriberdataservicecommands | Active | OK | Enterprise Application | 100 |
| subscriberdataservice(5.0.0.0.5.0.0.0.0) | Active | | Library | 100 |

Select **proxyregistrar** box to **Stop**.

This will move **proxyregistrar** from Active State to Prepared State.

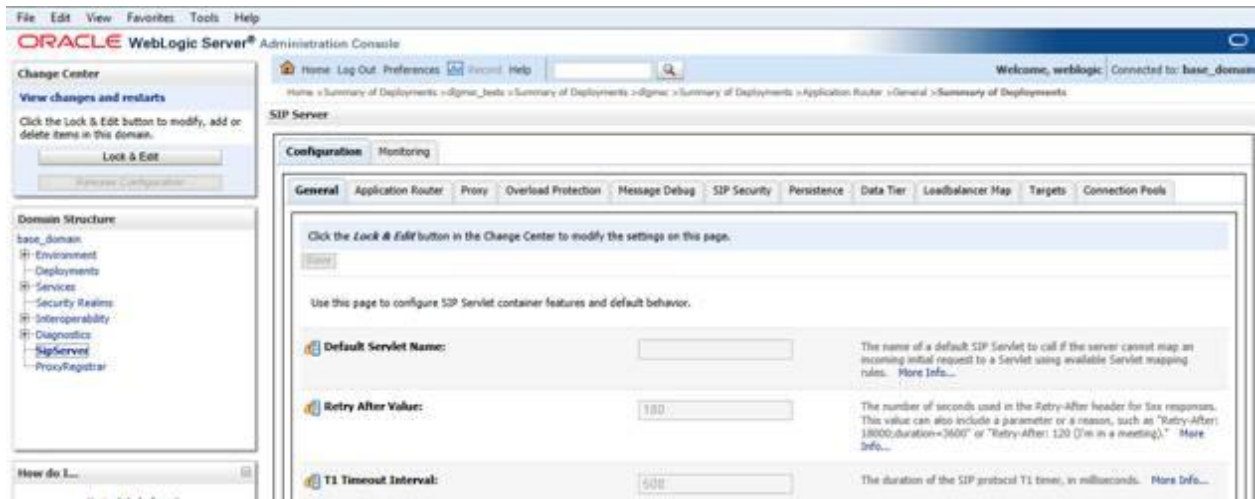


The screenshot shows the Oracle WebLogic Server Administration Console after the 'proxyregistrar' deployment has been stopped. The 'proxyregistrar' deployment is now in the 'Prepared' state. The table shows the following data:

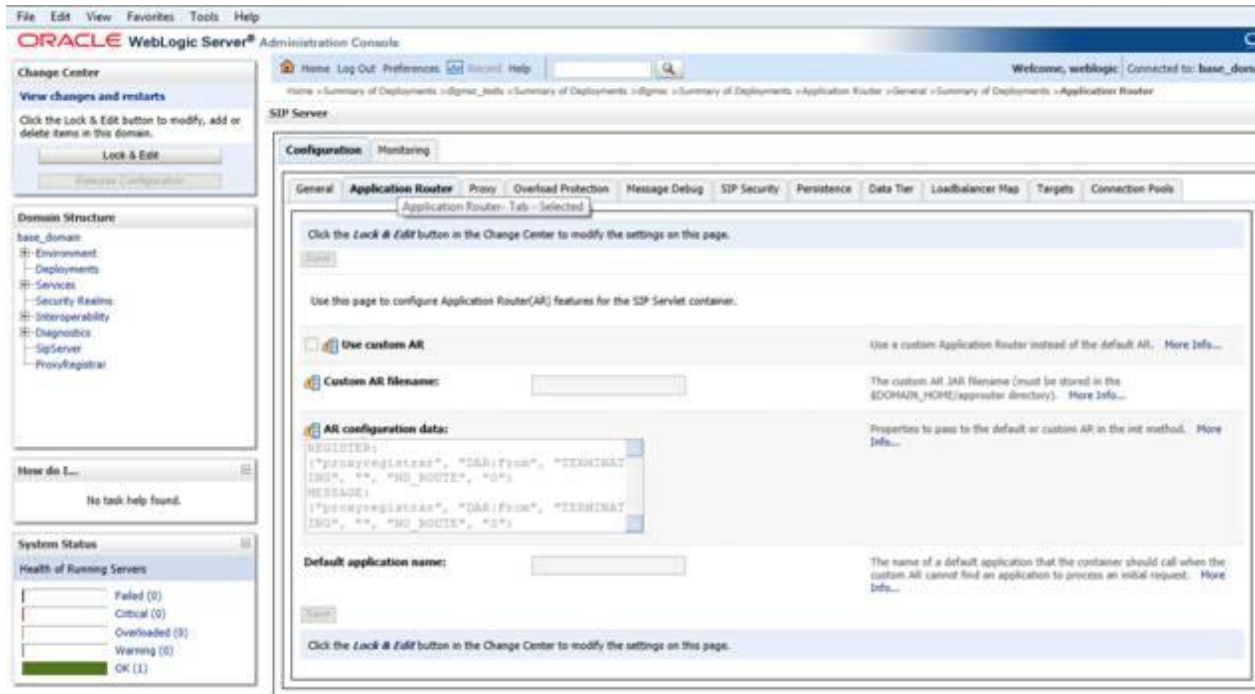
| Name | State | Health | Type | Deployment Order |
|--|--------|--------|------------------------|------------------|
| diagnostic | Active | OK | Web Application | 1 |
| diagnostic_tests | Active | OK | Web Application | 100 |
| oracle.sdp.platform(5.0.0.0.5.0.0.0.0) | Active | | Library | 100 |
| subscriberdataservicecommands | Active | OK | Enterprise Application | 100 |
| subscriberdataservice(5.0.0.0.5.0.0.0.0) | Active | | Library | 100 |

Select **proxyregistrar** box again then select **Delete**.

Click on **Activate Changes**.



Select **SipServer** under **Domain Structure**.



Select **Application Router** under **Configuration**.

Click on **Lock & Edit**.

Delete content from **AR configuration data** and click **Save** then **Activate Changes**.

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:

```
player@<OCCAS5-IP-ADDRESS>
```

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