Dialogic® PowerMedia™ XMS JSR 309 Connector Software Release 5.2

Installation and Configuration Guide
with Oracle Communications Converged Application Server 5.1

March 2016 Rev 1.0

www.dialogic.com

Copyright and Legal Notice

Table of Contents

Revision History

Revision	Release Date	Notes
1.0	March 2016	Initial release of this document.
Last modified: March 2016		

1. Dialogic JSR 309 Connector Requirements

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

- A functional Oracle Communications Converged Application Server 5.1 (OCCAS5.1) platform for development and testing.

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

  - Install the required Oracle patch:

    18135712 SU Patch [4Q2T]: SIP RP 5.1.0.0.2 - Do Async actions not always running if the action is created within a timer. (Patch)
- A functional PowerMedia XMS Release 3.1 system.
- SIP phones or soft clients.

Note: WebRTC is not supported by OCCAS5.1 platform.

OCCAS5.1 Application Server Platform	Dialogic JSR 309 Connector
occas510_ja_generic.jar	dialogic309-5.X.xxxx-occas5.1.tar

2. Contents of the Distribution

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

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

dialogic309-M.m.BBBB-occas5.1.tar

Where:

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

Note: OCCAS5.1 connector is built with Java version 1.6.

This package contains the following structure:

JSR 309 Connector Files	Description
DIR: */DlgcJSR309/application/* CONTENTS: *dlgc_sample_demo.war* *Dialogic.mp4* *Project/*	Directory that contains the Dialogic JSR 309 Verification Application *dlgc_sample_demo.war* ready to be deployed and the *Dialogic.mp4* media file used by the Verification Application (which will be part of upcoming PowerMedia XMS installs). Directory also contains the project directory that has all of the necessary items to build *dlgc_sample_demo.war*. Refer to Dialogic JSR 309 Verification Application for details.
DIR: */DlgcJSR309/dialogic309Connector/* */DlgcJSR309/3rdPartyLibs/*	Directory that contains the Dlgc309Connector directory, which has all of the Dialogic connector files, and the 3rdPartyLibs directory, which has all necessary third-party JAR files.
DIR: */DlgcJSR309/properties/* CONTENTS: *dlgc_sample_demo.properties* *log4j2.xml*	Directory that contains Verification Application properties files used to set up its configuration and the configuration parameters for Dialogic JSR 309 Connector Directory also contains the *Log4j2.xml* log configuration file used for Dialogic JSR 309 Connector and Verification Application logging.

3. Installation and Configuration

This section describes how to install and use the Dialogic JSR 309 Connector. The Dialogic JSR 309 Connector adds the Media Control API interface to an application running in a J2EE platform. The connector and the application need to be correctly configured on a platform for proper operation. This section contains the following procedures:

1. Preparing the J2EE Converged Application Server
2. Installing the Dialogic JSR 309 Connector
3. Installing the Dialogic JSR 309 Verification Application
4. Configuring the PowerMedia XMS Media File
5. Running the Dialogic JSR 309 Verification Application

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

Preparing the J2EE Converged Application Server

The Dialogic JSR 309 Connector has been deployed and tested on specific versions of OCCAS 5.1 as described in Dialogic JSR 309 Connector Requirements*.* For a quick guide on how to install and configure the desired Application Server (AS) before configuring the Dialogic JSR 309 Connector, refer to Appendix A: Dialogic JSR 309 Connector Environment Setup.

Installing the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is a library where, once properly installed and configured, it can be used by an application.

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

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

1. Configure the Application Server Platform
2. Install the Dialogic JSR 309 Connector

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

Configure the Application Server Platform

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

```
tar -xvf dialogic309-x.x.xxxx-occas5.1.tar
```

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

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

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

1. Set Up the Platform's Environment
2. Set Up and Configure the Logging Facility

Set Up the Platform's Environment

Edit the *startWeblogic.sh* file located here:

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

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

```
…..
JAVA_OPTIONS="${SAVE_JAVA_OPTIONS}"
SAVE_JAVA_OPTIONS=""
CLASSPATH="${SAVE_CLASSPATH}"
#Dialogic additions
LOG4J_OPTIONS="-Dlog4j.configurationFile=${DOMAIN_HOME}/config/Dialogic/log4j2.xml"
CLASSPATH="${DOMAIN_HOME}/lib/org.osgi-3.0.0.jar:${DOMAIN_HOME}/lib/log4j-api-
2.2.jar:${DOMAIN_HOME}/lib/log4j-slf4j-impl-2.2.jar:${DOMAIN_HOME}/lib/log4j-core-
2.2.jar:${DOMAIN_HOME}/lib/slf4j-api-1.7.5.jar:${CLASSPATH}"
SERIALIZATION="-Dwlss.local.serialization=false"
DLGC_OPTS="${SERIALIZATION} ${LOG4J_OPTIONS}"
# END Dialogic additions


SAVE_CLASSPATH=""
trap 'stopAll' 1 2 3 15
…..
………
echo "starting weblogic with Java version:"
${JAVA_HOME}/bin/java ${JAVA_VM} -version
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
        echo "Starting WLS with line:"
        echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME}
${DLGC_OPTS} -Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS}
${PROXY_SETTINGS} ${SERVER_CLASS}"
        ${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} -Dweblogic.Name=${SERVER_NAME} ${DLGC_OPTS}
-Djava.security.policy=${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} ${DLGC_OPTS}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy ${JAVA_OPTIONS} ${PROXY_SETTINGS}
${SERVER_CLASS}  >"${WLS_REDIRECT_LOG}" 2>&1
fi
stopAll
……..
```

Set Up and Configure the Logging Facility

The log4j2 logging facility is implemented through five third-party log4j2 library files (JAR files), which need to be placed in the appropriate platform *lib* directory. From Dialogic JSR 309 Connector distribution, copy the following five JAR files:

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

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

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

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

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

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

Note the following about the logging facility:

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

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

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

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

Install the Dialogic JSR 309 Connector

The Dialogic JSR 309 Connector is distributed as a set of library (JAR) files. The set includes the following files:

- Set of Dialogic JSR 309 Connector library (JAR) files:
  - *dialogic309-5.0.xxxx-occas5.1.jar*
  - *dialogicmsmltypes-5.0-xxxx.jar*
  - *dialogicsmiltypes-5.0-xxxx.jar*
- Set of required third-party library (JAR) files:
  - *geronimo-commonj_1.1_spec-1.0.jar*
  - *jain-sip-sdp-1.2.91.jar*
  - *jsr173_1.0_api.jar*
  - *xbean.jar*

From distribution, copy Dialogic JSR 309 Connector and third-party JAR files as referenced above into following platform directory:

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

The Dialogic JSR 309 Connector is now properly installed and ready to be used by an application.

Installing the Dialogic JSR 309 Verification Application

Once the Dialogic JSR 309 Connector is installed, an application can take advantage of the Dialogic JSR 309 Connector and use its resources for media related functionality. The Dialogic JSR 309 Connector package provides a demo application that uses the Dialogic JSR 309 Connector to illustrate various media functionalities.

The following procedures illustrate how to install and configure the Dialogic JSR 309 Verification Application and how to verify that the Verification Application works with JSR 309 Connector and communicates correctly with PowerMedia XMS.

1. Configure the Dialogic JSR 309 Verification Application Properties File
2. Set Up Environment Variables
3. Install the Dialogic JSR 309 Verification Application

Configure the Dialogic JSR 309 Verification Application Properties File

Dialogic JSR 309 Verification Application uses a properties file to configure itself and the Dialogic JSR 309 Connector. From the distribution package under *DlgcJSR309/properties*, copy *dlgc_sample_demo.properties* file to the following location:

```
${DOMAIN_HOME}\config\Dialogic\dlgc_sample_demo.properties
```

Edit the properties file. Modify the highlighted fields in **RED** to fit your environment:

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

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

Note the following:

- *dlgc.jsr309.driver.name* is a fixed name that the application will reference when loading the Dialogic JSR 309 Connector factory.
- *connector.sip.address* is a platform SIP IP address used by the SIP container. The connector provides the ability to change the address in case the platform has

multiple IP interfaces and the default IP address picked by the connector needs to be changed.

- *connector.sip.port* is a platform SIP port address used by the SIP container. The connector provides the ability to change the address in case the platform has multiple IP interfaces and the proper one is defined for a different port number.
- *mediaserver.sip.ipaddress* is a Dialogic PowerMedia XMS Media Server SIP IP address to be used by the Dialogic JSR 309 Connector.
- *mediaserver.sip.port* is a Dialogic PowerMedia XMS Media Server SIP port to be used by the Dialogic JSR 309 Connector.

Set Up Environment Variables

In this example, the OCCAS 5.1 Application Server platform was used, so the user "occas5.1" was created on the Linux system. The file to be edited will be found in this user's root directory (i.e., */home/occas5.1*).

Edit the system user's .bashrc file:

```
vi .bashrc
```

Add the following lines marked in RED as below:

```
# .bashrc

export JAVA_HOME=/usr/java/jdk1.6.0_45

### Dialogic additions
export DOMAIN_HOME=/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain
export SAMPLE_PROPERTY_FILE=${DOMAIN_HOME}/config/Dialogic/dlgc_sample_demo.properties
### END - Dialogic additions

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

# User specific aliases and functions
```

Note the following about Dialogic additions:

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

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

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

```
source /home/occas5.1/.bashrc
```

Install the Dialogic JSR 309 Verification Application

Make sure that JSR 309 Verification Application WAR file, which is provided with distribution, exists in OCCAS under <*Domain Location*>/*application* directory.

Then, access the Administration Console through the web browser at:

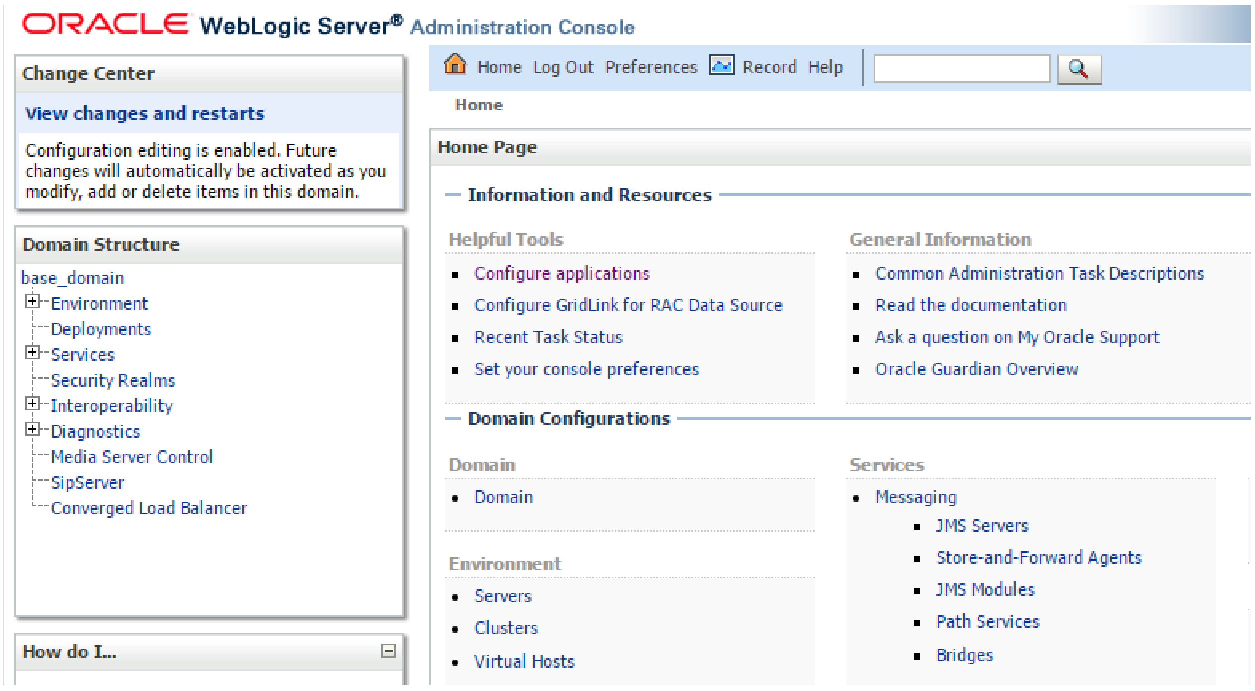
http://<as_ip_address>:7001/console

Use the username/password that was set during configuration. The following is an example:

Name: weblogic

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

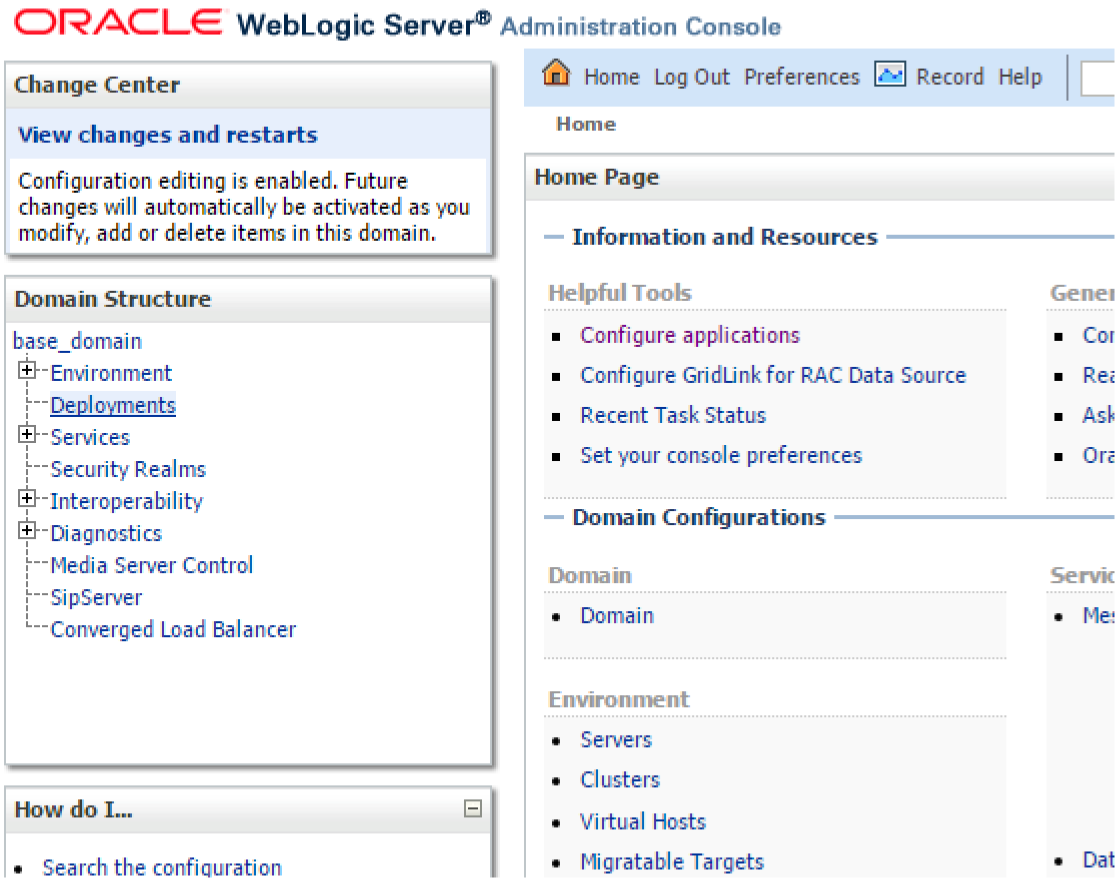
Note: The password is defined during OCCAS installation.



Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Install the Verification Application as follows:

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



ORACLE WebLogic Server® Administration Console

Home Log Out Preferences Record Help

Home

Change Center

View changes and restarts

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Domain Structure

base_domain
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics
- Media Server Control
- SipServer
- Converged Load Balancer

How do I...

- Search the configuration

Home Page

Information and Resources

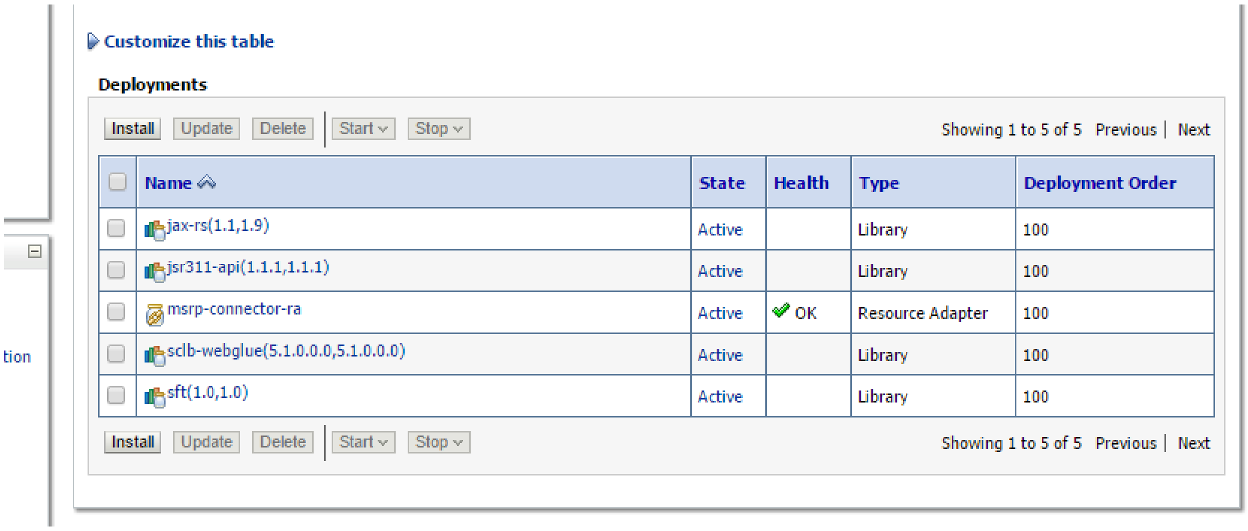
Helpful Tools
- Configure applications
- Configure GridLink for RAC Data Source
- Recent Task Status
- Set your console preferences

Domain Configurations

Domain
- Domain

Environment
- Servers
- Clusters
- Virtual Hosts
- Migratable Targets

2. Make sure the existing services are displayed to verify that the OCCAS components have started.



Customize this table

Deployments

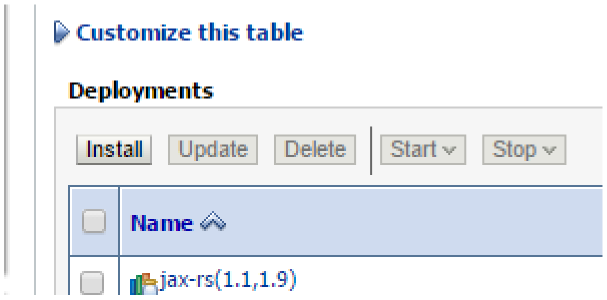
Install Update Delete Start Stop — Showing 1 to 5 of 5 Previous | Next

	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

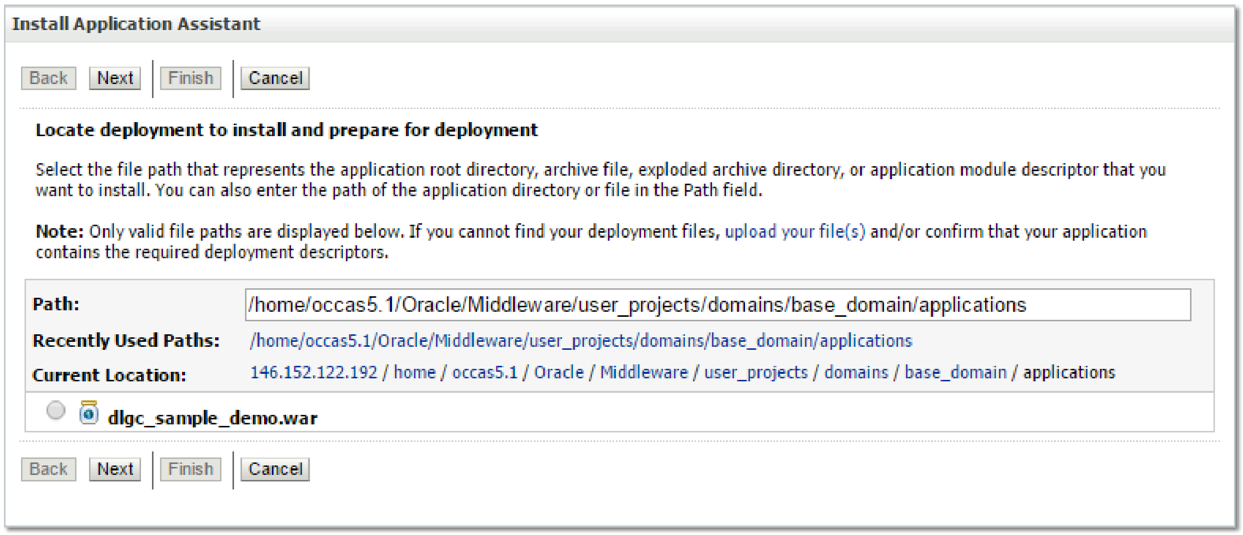
Install Update Delete Start Stop — Showing 1 to 5 of 5 Previous | Next

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

3. Click **Install**.



4. In the **Current Location** field of the **Install Application Assistant** screen, navigate to *<Domain Location>/applications*, select **dlgc_sample_demo.war**, and then click **Next**.



Locate deployment to install and prepare for deployment

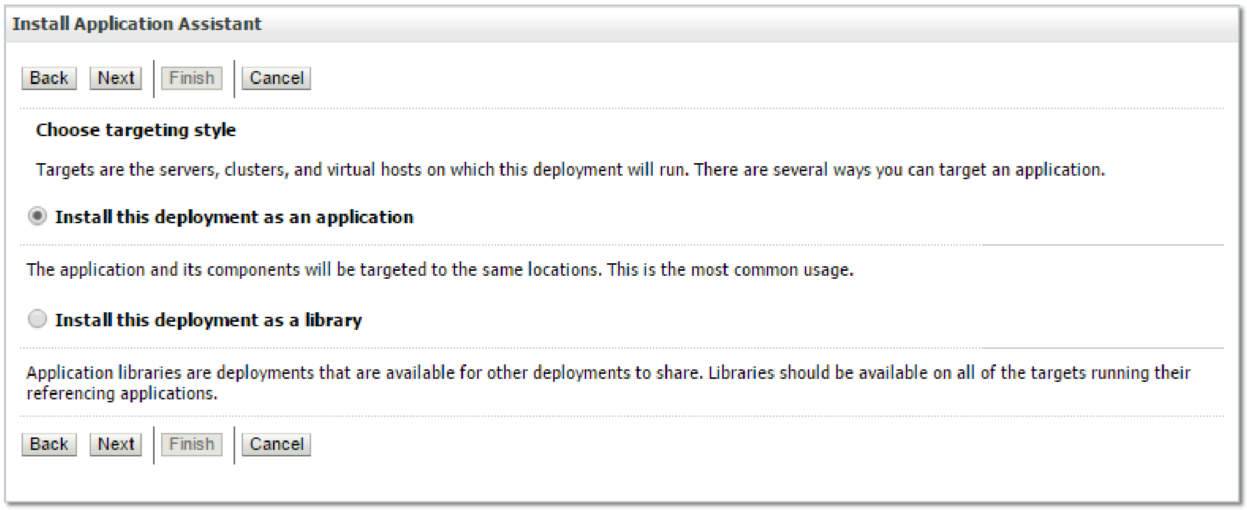
Select the file path that represents the application root directory, archive file, exploded archive directory, or application module descriptor that you want to install. You can also enter the path of the application directory or file in the Path field.

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:	/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/applications
Recently Used Paths:	/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/applications
Current Location:	146.152.122.192 / home / occas5.1 / Oracle / Middleware / user_projects / domains / base_domain / applications

dlgc_sample_demo.war

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



Choose targeting style

Targets are the servers, clusters, and virtual hosts on which this deployment will run. There are several ways you can target an application.

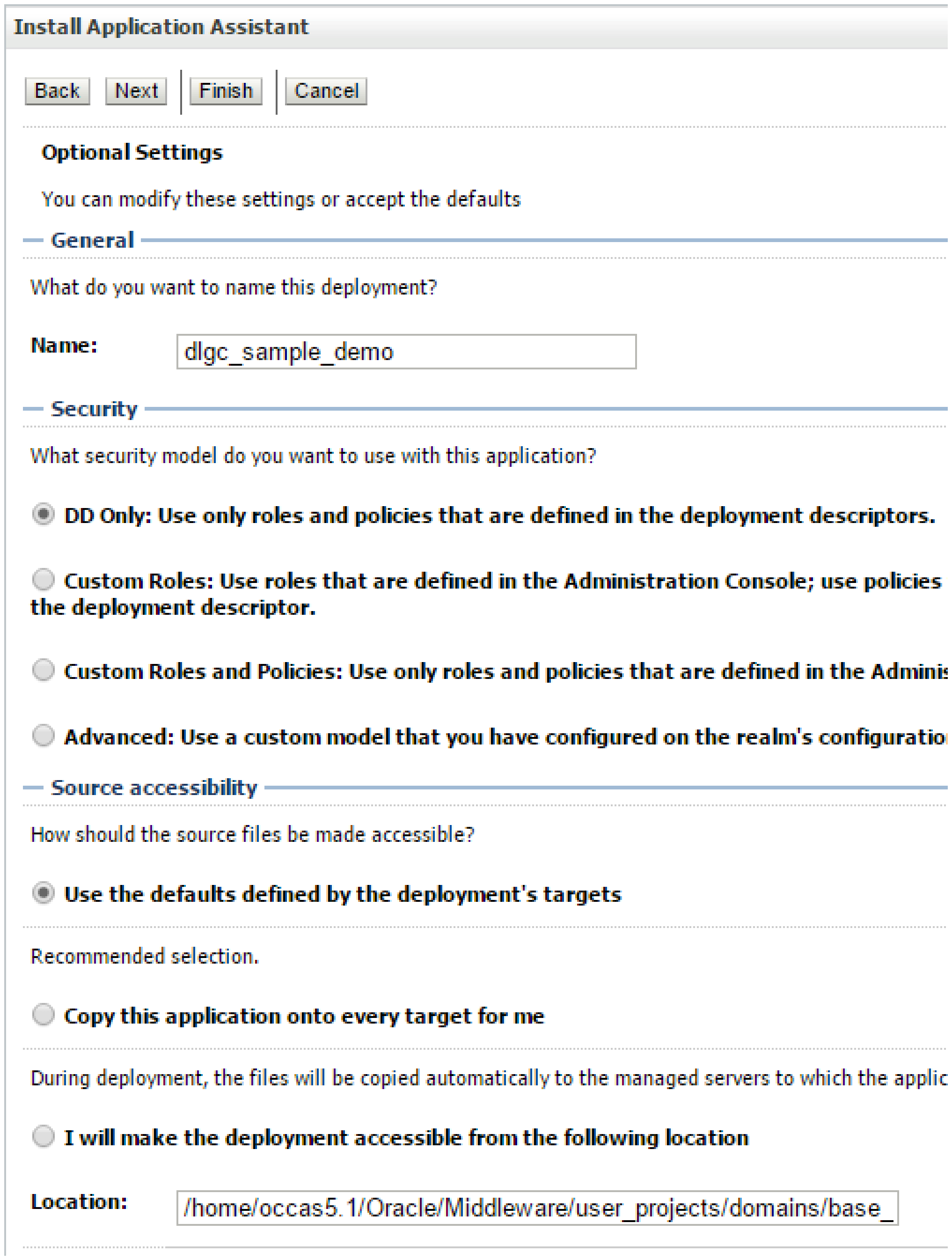
● Install this deployment as an application

The application and its components will be targeted to the same locations. This is the most common usage.

○ Install this deployment as a library

Application libraries are deployments that are available for other deployments to share. Libraries should be available on all of the targets running their referencing applications.

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



Optional Settings

You can modify these settings or accept the defaults

General

What do you want to name this deployment?

Name: dlgc_sample_demo

Security

What security model do you want to use with this application?

[✓ **DD Only: Use only roles and policies that are defined in the deployment descriptors.**]

Custom Roles: Use roles that are defined in the Administration Console; use policies the deployment descriptor.

Custom Roles and Policies: Use only roles and policies that are defined in the Adminis

Advanced: Use a custom model that you have configured on the realm's configuratio

Source accessibility

How should the source files be made accessible?

[✓ **Use the defaults defined by the deployment's targets**]

Recommended selection.

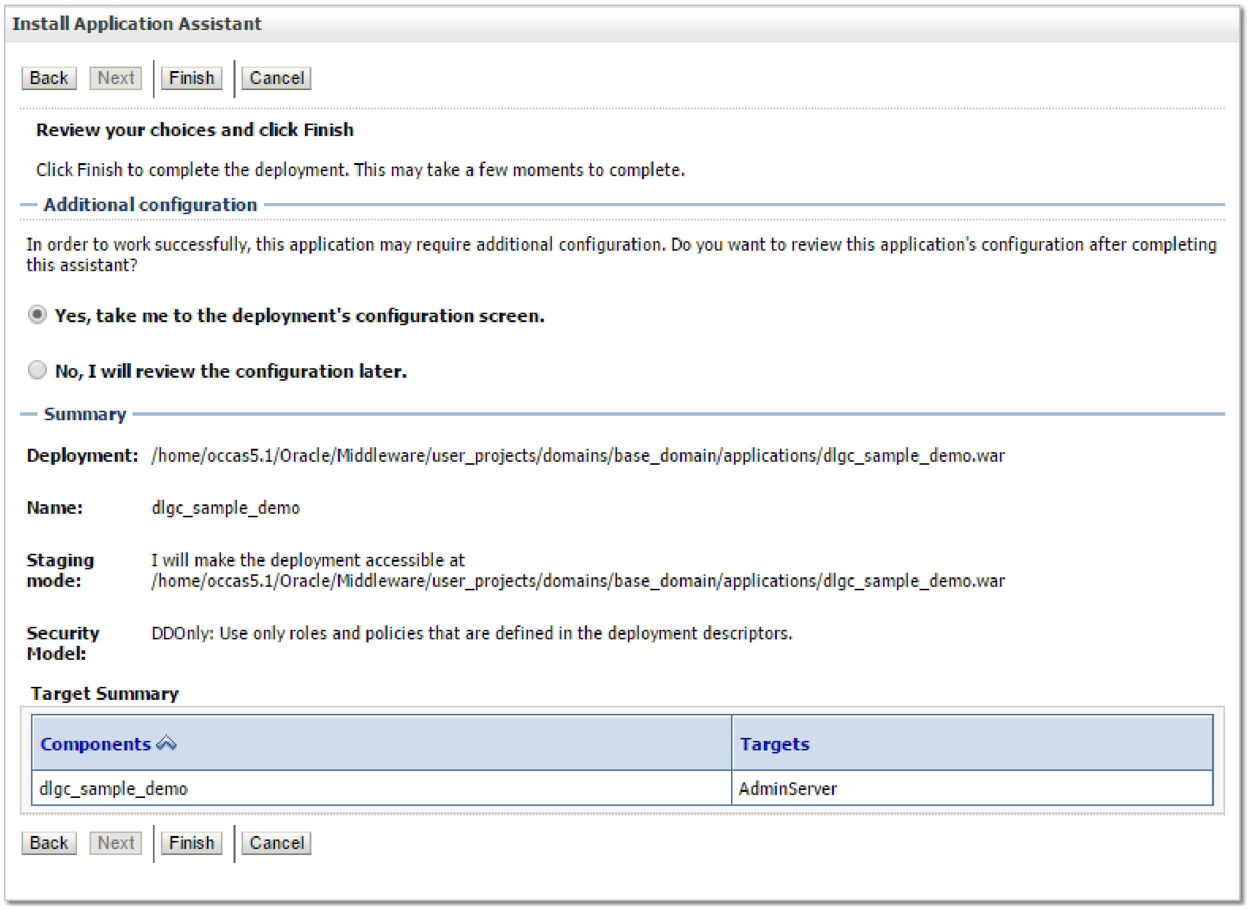
Copy this application onto every target for me

During deployment, the files will be copied automatically to the managed servers to which the applic

I will make the deployment accessible from the following location

Location: /home/occas5.1/Oracle/Middleware/user_projects/domains/base_

7. Click **Finish**.



Install Application Assistant

Back Next Finish Cancel

Review your choices and click Finish

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

Additional configuration

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

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

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

Summary

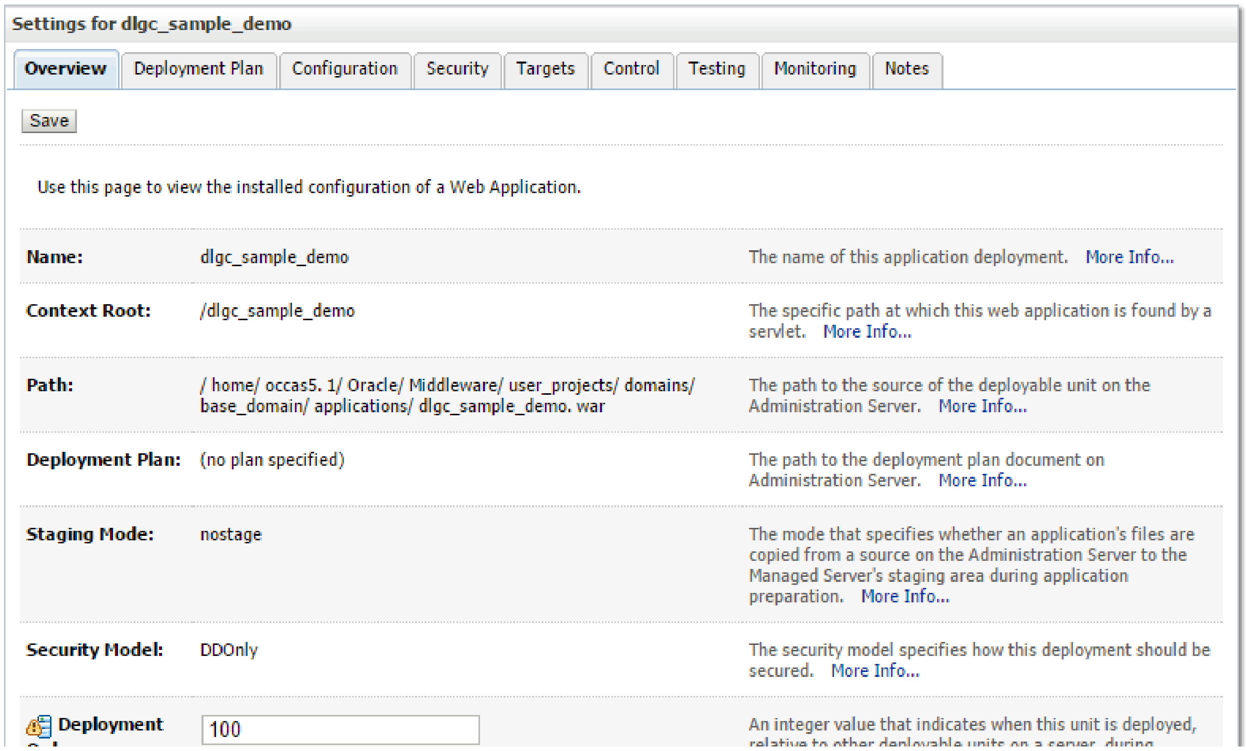
Deployment:	/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/applications/dlgc_sample_demo.war
Name:	dlgc_sample_demo
Staging mode:	I will make the deployment accessible at /home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/applications/dlgc_sample_demo.war
Security Model:	DDOnly: Use only roles and policies that are defined in the deployment descriptors.

Target Summary

Components	Targets
dlgc_sample_demo	AdminServer

Back Next Finish Cancel

8. Click **Save**.



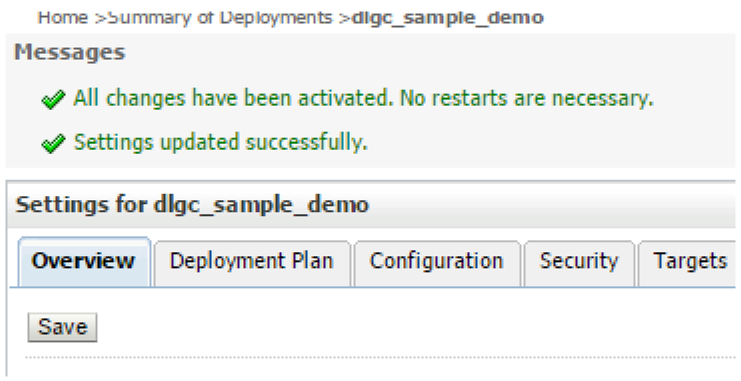
Settings for dlgc_sample_demo

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

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

Name:	dlgc_sample_demo	The name of this application deployment. More Info...
Context Root:	/dlgc_sample_demo	The specific path at which this web application is found by a servlet. More Info...
Path:	/ home/ occas5. 1/ Oracle/ Middleware/ user_projects/ domains/ base_domain/ applications/ dlgc_sample_demo. 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	100	An integer value that indicates when this unit is deployed,

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



Home >Summary of Deployments >**dlgc_sample_demo**

Messages

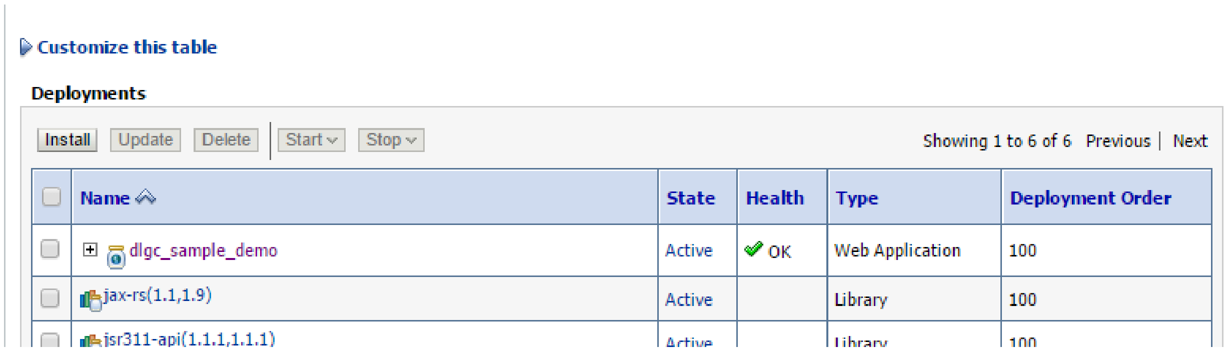
✓ All changes have been activated. No restarts are necessary.

✓ Settings updated successfully.

Settings for dlgc_sample_demo

Overview | Deployment Plan | Configuration | Security | Targets
Save

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



Deployments

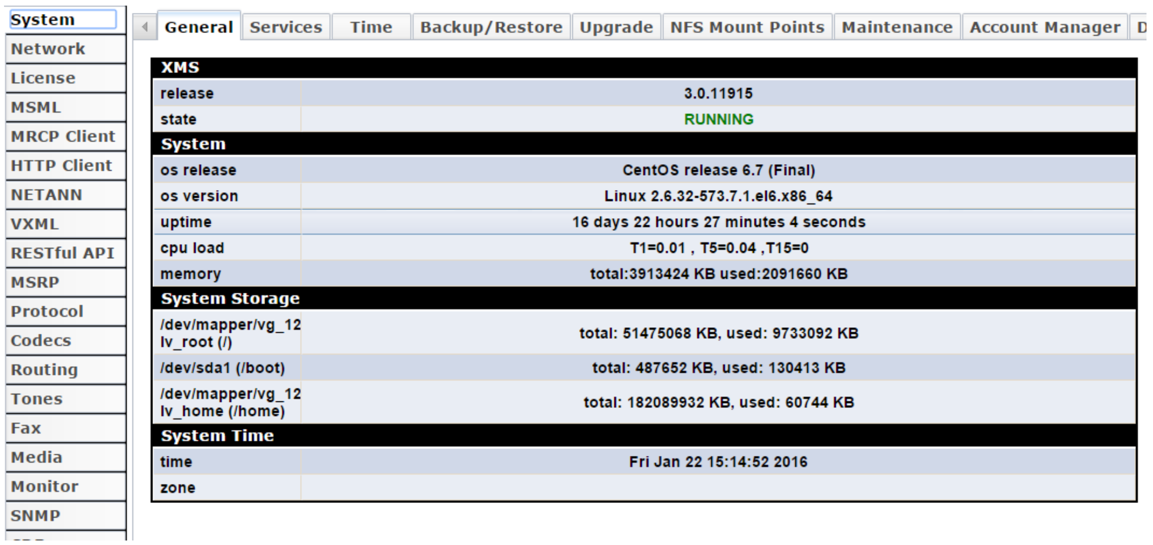
Showing 1 to 6 of 6 Previous | Next

	Name	State	Health	Type	Deployment Order
☐	⊞ dlgc_sample_demo	Active	✓ OK	Web Application	100
☐	jax-rs(1.1,1.9)	Active		Library	100
☐	jsr311-api(1.1.1,1.1.1)	Active		Library	100

Configuring the PowerMedia XMS Media File

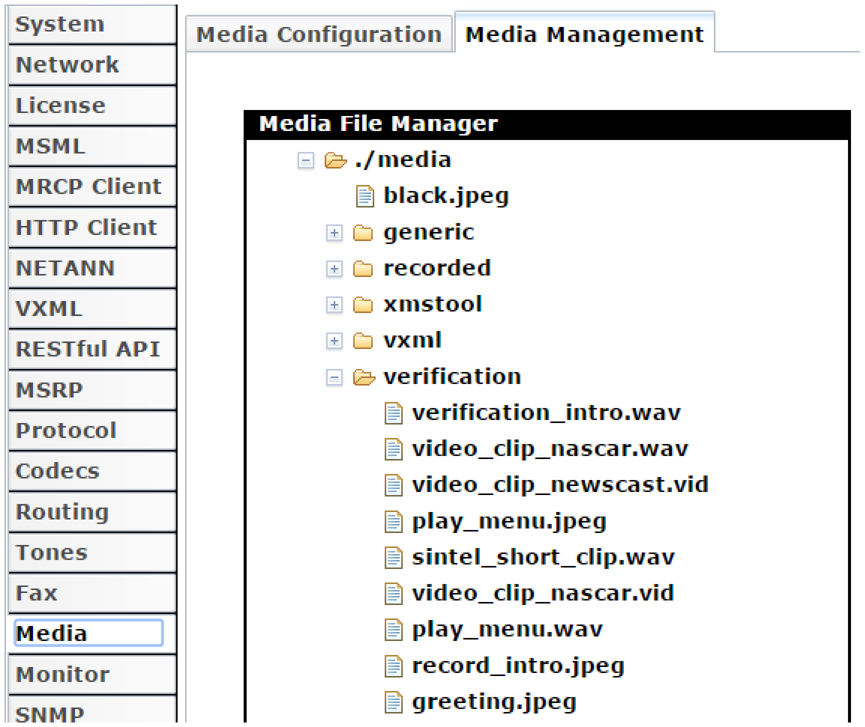
The Dialogic JSR 309 Verification Application has been developed to use the *Dialogic.mp4* media file. This media file will become part of Dialogic PowerMedia XMS distribution; however, as of PowerMedia XMS release 3.0 Service Update 1, the media file is not part of distribution and must be installed manually.

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



XMS	
release	3.0.11915
state	RUNNING
System	
os release	CentOS release 6.7 (Final)
os version	Linux 2.6.32-573.7.1.el6.x86_64
uptime	16 days 22 hours 27 minutes 4 seconds
cpu load	T1=0.01 , T5=0.04 ,T15=0
memory	total:3913424 KB used:2091660 KB
System Storage	
/dev/mapper/vg_12 lv_root (/)	total: 51475068 KB, used: 9733092 KB
/dev/sda1 (/boot)	total: 487652 KB, used: 130413 KB
/dev/mapper/vg_12 lv_home (/home)	total: 182089932 KB, used: 60744 KB
System Time	
time	Fri Jan 22 15:14:52 2016
zone	

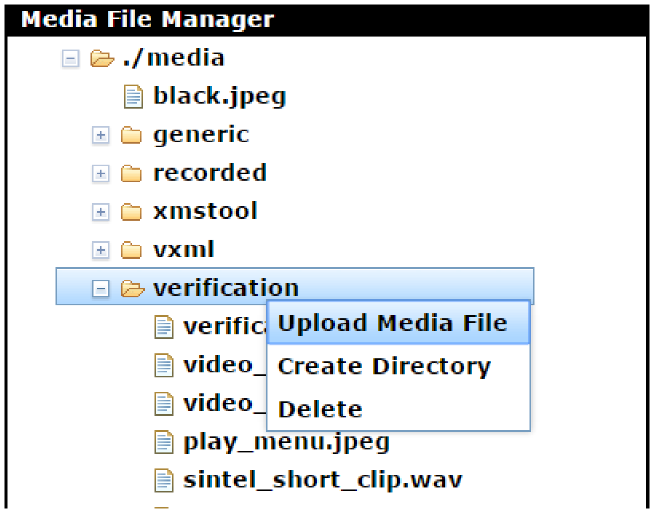
2. Click the **Media Management** tab.



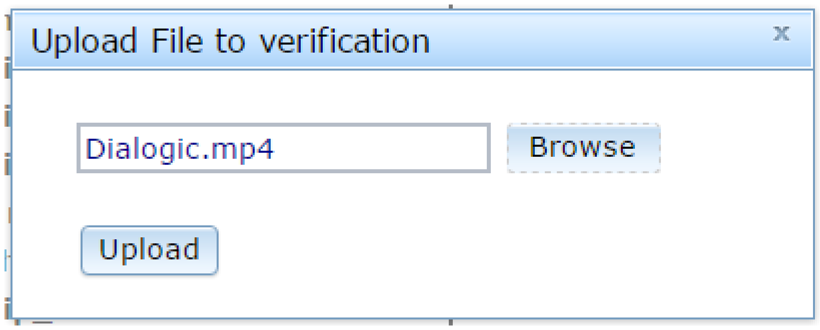
Media File Manager

```
./media
    black.jpeg
    generic
    recorded
    xmstool
    vxml
    verification
        verification_intro.wav
        video_clip_nascar.wav
        video_clip_newscast.vid
        play_menu.jpeg
        sintel_short_clip.wav
        video_clip_nascar.vid
        play_menu.wav
        record_intro.jpeg
        greeting.jpeg
```

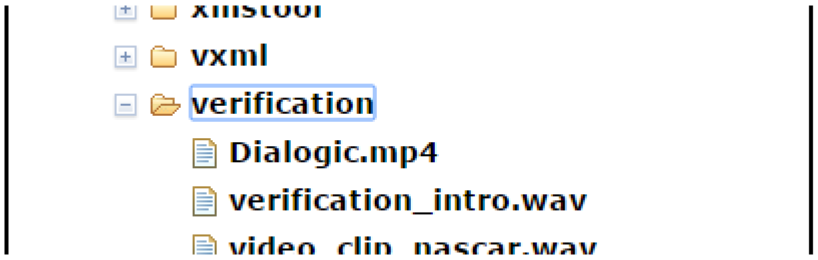
3. In the **Media File Manager** section, Right-click the **verification** folder and click **Upload Media File**.



4. Click **Browse**, select *Dialogic.mp4* in the Dialogic JSR 309 Connector distribution folder, and click **Upload**.



Once uploaded, the *Dialogic.mp4* media file is stored in the appropriate location for the Dialogic JSR 309 Verification Application to use it as per its configuration.



Running the Dialogic JSR 309 Verification Application

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

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

Perform the following procedure for SIP client verification:

1. Configure a SIP client for the supported audio/video codec.
2. Call into the OCCAS 5.1 Application Server with following URI:

   ```
   player@<AS_ip_address>
   ```

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

4. Dialogic JSR 309 Verification Application

About

The Dialogic JSR 309 Verification Application is provided with each platform-specific package for two reasons:

1. The application (WAR file), which uses the Dialogic JSR 309 Connector, is provided as a tool to verify the Application Server platform and Dialogic PowerMedia XMS operation.
2. The application project source has all the necessary components required to create a platform-specific application using the Dialogic JSR 309 Connector. This can quickly help clarify various steps that are required in the J2EE application using the Dialogic JSR 309 Connector. It includes the following:
    a. Provides steps on how to create an application (WAR file) to run in a specific J2EE AS platform
    b. Illustrates application initialization steps
    c. Illustrates application initialization steps necessary for use with the Dialogic JSR 309 Connector
    d. Illustrates the steps the application needs to take in order to work with SIP clients

The Details

This section details the different areas of the Verification Application for better understanding of the basic, necessary steps for any application.

- Application WAR File Content
- Application Initialization
- Mapping SIP Traffic to the Application
- Dialogic JSR 309 Connector Initialization

Application WAR File Content

As an example, the content of the application is illustrated in the Dialogic JSR 309 Verification Application WAR file. The WAR package contains several necessary items. Refer to *build.xml* to get familiar with how the WAR file is generated.

The *dlgc_sample_demo.war* file consists of two directories:

- The */META-INF* directory contains a *MANIFEST.MF,* which a standard way of providing information about the package that contains it.
- The */WEB-INF* directory contains the following:
    - The *classes* directory, which contains java .class files.
    - The *lib* directory, which contains all JAR files required by the deployment application WAR file.
    - The *sip.xml* deployment descriptor used by the SIP servlet container to process deployed SIP applications and configure the runtime to properly respond to incoming SIP requests

Application Initialization

Below is an example of a basic application structure used in this platform.

```
package play;

public class AsyncPlayer extends SipServlet implements Serializable, SipServletListener
{
        @Override
        public void init(ServletConfig cfg) throws ServletException
        {
        }

        @Override
        public void servletInitialized(SipServletContextEvent evt)
        {
        }
```

The AsyncPlayer class extends Sip Servlets, which are called when the appropriate SIP bound incoming call is received by the application platform.

When the platform starts the application, it will invoke an init( ) function. This function should contain application specific initialization procedures. This is where the Verification Application reads the application properties file and stores its content in local storage to be used later when initializing the JSR 309 interface.

Then, once the platform's SIP container is started, it will call the application's servletInitialized( ) method to inform it that the SIP stack is now ready for application usage. At this stage, the application can start to initialize the Dialogic JSR 309 Connector.

Mapping SIP Traffic to the Application

This example provides information on the *sip.xml* content, which is used by the application platform to map SIP requests to a desired application SIP Servlet. The example is broken into four sections.

1. This first section defines mandatory *sip.xml* content so it can be correctly initialized by the application platform. *app-name* and *javaee:display-name* define the application name and can be any string that describes the application.

```
<?xml version="1.0" encoding="UTF-8"?>
<sip-app id="SipApp_ID"  xmlns:javaee="http://java.sun.com/xml/ns/javaee"
xmlns="http://www.jcp.org/xml/ns/sipservlet" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:schemaLocation="http://www.jcp.org/xml/ns/sipservlet
http://www.jcp.org/xml/ns/sipservlet/sip-app_1_1.xsd http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <app-name>Dialogic-Samples</app-name>
  <javaee:display-name>Dialogic-Samples</javaee:display-name>

  <session-config>
   <javaee:session-timeout>0</javaee:session-timeout>
  </session-config>
…
```

2. This section illustrates the mandatory <*servlet*> section that needs to be defined by an application to properly define Dialogic JSR 309 Connector SIP Servlet. Since the connector is a library that is included as part of an application, the application sip.xml needs to define necessary entries required by the Dialogic JSR 309 Connector. The content in this section is fixed because this is driven by Dialogic JSR 309 Connector implementation.

   Note: *javaee:load-on-startup* is set to 1. This startup order number needs to be smaller than the application using the Dialogic JSR 309 Connector.

```
  <servlet>
    <javaee:description>Dialogic JSR 309 Connector SIP Servlet component talking to
Dialogic XMS</javaee:description>
    <javaee:display-name>DlgcSipServlet</javaee:display-name>
    <javaee:servlet-name>DlgcSipServlet</javaee:servlet-name>
    <javaee:servlet-
class>com.vendor.dialogic.javax.media.mscontrol.sip.DlgcSipServlet</javaee:servlet-class>
    <javaee:load-on-startup>1</javaee:load-on-startup>
  </servlet>
…
```

3. This section illustrates the mandatory <servlet> and <listener> pair that is used by the application. Note the following:
   - *javaee:description* – String describing the application.
   - *javaee:display-name* – String displaying the application name.
   - *javaee:servlet-name* – String matching application servlet name.
   - *javaee:load-on-startup* – Order number to start the application’s servlet.

     Important: Since this application is using the Dialogic JSR 309 Connector, the order number is greater than the connector startup order number.
   - *javaee:servlet-class* – Points to the <package name>.<listener class>.

```
…
  <servlet>
    <javaee:description>Simple Player Test</javaee:description>
    <javaee:display-name>AsyncPlayer</javaee:display-name>
    <javaee:servlet-name>AsyncPlayer</javaee:servlet-name>
    <javaee:load-on-startup>2</javaee:load-on-startup>
    <javaee:servlet-class>play.AsyncPlayer</javaee:servlet-class>
  </servlet>

  <listener>
    <javaee:description>AsyncPlayer</javaee:description>
    <javaee:display-name>AsyncPlayer</javaee:display-name>
    <javaee:listener-class>play.AsyncPlayer</javaee:listener-class>
```

4. This next section defines mapping of SIP requests to appropriate servlets. The *<servlet-mapping>* section needs to be specified since it is required for the Dialogic JSR 309 Connector. The *<servlet-mapping>* section defines the user part of the URI that, when received by the platform, will be delivered to the appropriate servlet.
    - *<servlet-name>* - Defines the servlet class name to deliver the incoming request to.
    - *<value>* - Defines what user part of URI to identify the incoming request to be handled by this application's servlet.

```
…
  <servlet-selection>
    <servlet-mapping>
      <servlet-name>DlgcSipServlet</servlet-name>
      <pattern>
        <and>
          <equal>
            <var>request.method</var>
            <value>INVITE</value>
          </equal>
          <equal>
            <var>request.to.uri.user</var>
            <value></value>
          </equal>
        </and>
      </pattern>
    </servlet-mapping>
    <servlet-mapping>
      <servlet-name>AsyncPlayer</servlet-name>
      <pattern>
        <and>
          <equal>
            <var>request.method</var>
            <value>INVITE</value>
          </equal>
          <equal>
            <var>request.to.uri.user</var>
            <value>player</value>
          </equal>
        </and>
      </pattern>
    </servlet-mapping>
  </servlet-selection>
</sip-app>
```

Dialogic JSR 309 Connector Initialization

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

In this servletInitialized() method, the application calls the initDriver() method, which obtains the Dialogic JSR 309 Connector automatic configuration.

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

The connector driver is able to discover some of the parameters that it needs but not all. The parameters required by the driver to work correctly are as follows:

- connector.sip.address – Platform SIP IP address used by the SIP container. The connector provides the ability to change the address in case the platform has multiple IP interfaces and the default IP address picked by connector needs to be changed.
- connector.sip.port - Platform SIP port address used by SIP container. The connector provides ability to change the address in case the platform has multiple IP interfaces and the proper one is defined for different port number.
- connector.sip.transport – Platform SIP transport. Supported values are "udp" or "tcp". Default: "udp".
- mediaserver.sip.ipaddress – Dialogic XMS Media Server SIP IP address to be used by the Dialogic JSR 309 Connector.
- mediaserver.sip.port - Dialogic XMS Media Server SIP port to be used by the Dialogic JSR 309 Connector.

Optionally, the Dialogic JSR 309 Connector supports turning on SIP Session Timers between the JSR 309 driver and the Dialogic PowerMedia XMS. In this version of the JSR 309 Connector, the SIP Session Timers are turned on by default. The application can modify the parameters for the SIP Session Timers when configuring factory properties:

- mediaserver.sessionTimer.maxTimeout – defines SIP Session timeout in seconds. Default: 1800 (seconds).
- mediaserver.sessionTimer.switch - Turns the SIP Session Timer function on or off. Allowed values are: "on" or "off". Default: "on".

The Verification Application creates new connector properties by taking information from defines in the application properties file to be used later when initializing the connector configuration.

```
…
  Properties factoryProperties = new Properties();
  for ( PropertyInfo prop: connectorProperty ) {
    log.debug("initDriver() - ===================");
    log.debug("initDriver() - Name: " + prop.name);
    log.debug("initDriver() - Description: " + prop.description);
    log.debug("initDriver() - Required: " + new Boolean(prop.required).toString() );
    log.debug("initDriver() - Value: " + prop.defaultValue);
    if ( prop.name.compareToIgnoreCase("connector.sip.address") == 0 )
    {
      if (prop.defaultValue.compareToIgnoreCase(new_connector_sip_address.toString()) != 0)
      {
        log.debug("initDriver() - New Value: " + new_connector_sip_address);
        prop.defaultValue = new_connector_sip_address;
      }
……
    factoryProperties.setProperty(prop.name, prop.defaultValue);
  }
….
```

The application creates a new properties factory in which it will store all required parameters for the Dialogic JSR 309 Connector to start properly. It reads the locally stored application properties file configuration of each required parameter and compares it to the value automatically picked up by the Dialogic JSR 309 Connector. It then takes the newest value for each of the required parameters and stores it in new properties factory.

The Dialogic JSR 309 Connector factory is created with the new set of parameters.

```
....
  mscFactory = dlgcDriver.getFactory(factoryProperties);
….
```

The Dialogic JSR 309 Connector factory (mscFactory) is now created with a new set of required parameters. Now, the Dialogic JSR 309 Connector interface can be used.

5. Troubleshooting

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

Logging

The Dialogic JSR 309 Connector and its Verification Application use the Apache Log4j2 (version 2) logging facility. The connector makes use of *log4j2.xml* file for its logging configuration. With the introduction of Log4j2, it is possible to change the log levels without stopping/restarting any components. All that needs to be done is open *log4j2.xml* and change the logging to the desired level. Log configuration file *log4j2.xml* can be found at:

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

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

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration monitorInterval="10" status="ERROR">
  <Appenders>
    <File name="dialogic" fileName="logs/Dialogic.log" append="false">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"/>
    </File>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} %-5level %class{36} %L %M - %msg%xEx%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <Logger name="com.vendor.dialogic" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="play" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
    <Logger name="base" level="ERROR">
      <AppenderRef ref="dialogic"/>
      <!-- AppenderRef ref="STDOUT"/ -->
    </Logger>
  </Loggers>
</Configuration>
```

For details of the *Log4j2.xml* configuration, refer to the following information:

- monitorInterval – Parameter defines how often log4j2 facility will automatically detect changes to the configuration file and reconfigure itself. The default is 10 seconds.
    - Appenders:
        - Provided *log4j2.xml* file defines two streams (Appenders) that it will send logging to: a file (*Dialogic.log*) and a system console. Each individual logger has a choice of which appender to use.
    - Loggers:
        - Provided *log4j2.xml* file Loggers section provides a logger configuration for various Java source packages:
            - com.vendor.dialogic is a Dialogic JSR 309 Connector.
            - play & base is a Dialogic JSR 309 Verification Application.

Note: Each logger can be set individually to the appropriate level of logging and each logger can be individually configured to log to file, STDOUT, or both.

Note that default logging level is set to *ERROR,* which will cause the *Dialogic.log* file to be empty unless there are errors.

Refer to the Apache Log4j 2 documentation at http://logging.apache.org/log4j/2.x for details.

Additional platform component logging, configuration, and modifications can be accomplished via appropriate Application Server Administration page. Refer to the platform specific documentation for details.

Dialogic JSR 309 Connector and Verification Application Troubleshooting

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

```
2016-01-15 12:33:27,045 INFO  [[ACTIVE] ExecuteThread: '3' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:60) -
LoadProperties() - Properties File =
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/config/Dialogic/dlgc_s
ample_demo.properties

2016-01-15 12:33:27,045 ERROR [[ACTIVE] ExecuteThread: '3' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:77) -
java.io.FileNotFoundException:
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/config/Dialogic/dlgc_s
ample_demo.properties (No such file or directory)

java.io.FileNotFoundException:
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/config/Dialogic/dlgc_s
ample_demo.properties (No such file or directory)
```

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

```
2016-01-15 13:46:14,054 INFO  [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] play.AsyncPlayer (AsyncPlayer.java:132) - init()
- Entering

2016-01-15 13:46:14,054 DEBUG [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] play.AsyncPlayer (AsyncPlayer.java:134) - init()
- servletName: AsyncPlayer

2016-01-15 13:46:14,055 INFO  [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:40) -
ConfigProperty() - Entering

2016-01-15 13:46:14,055 INFO  [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:56) -
LoadProperties() - Entering

2016-01-15 13:46:14,056 INFO  [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:60) -
LoadProperties() - Properties File =
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/config/Dialogic/dlgc_s
ample_demo.properties

2016-01-15 13:46:14,056 INFO  [[ACTIVE] ExecuteThread: '4' for queue:
'weblogic.kernel.Default (self-tuning)'] base.ConfigProperty (ConfigProperty.java:73) -
LoadProperties() - base Configuration File:
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/config/Dialogic/dlgc_s
ample_demo.properties Successfully Loaded
```

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

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

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

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

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

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

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

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

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

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

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

SIP Errors

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

1. Verify the available PowerMedia XMS licenses.
2. Check the */etc/hosts* file configuration.

6. Building and Debugging Sample Demos in Eclipse IDE

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

Prerequisites

The following components must be installed:

- JDK 1.6.0_45

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

- Eclipse KDE (Eclipse Standard SDK – Kepler Service Release 2 used here).
- In order to build the provided Verification Application, you will need to obtain three OCCAS 5.1.0 libraries, which are NOT provided with JSR 309 Connector distribution:
  - *javax.servlet_1.0.0.0_2-5.jar*
  - *sipservlet.jar*
  - *mscontrol.jar*
- Dialogic JSR 309 Connector package, three library (JAR) files will also be needed:
  - *dialogic309-5.0.xxxx-occas5.1.jar*
  - *dialogicmsmltypes-5.0-xxxx.jar*
  - *dialogicsmiltypes-5.0-xxxx.jar*

Creating the Build Environment

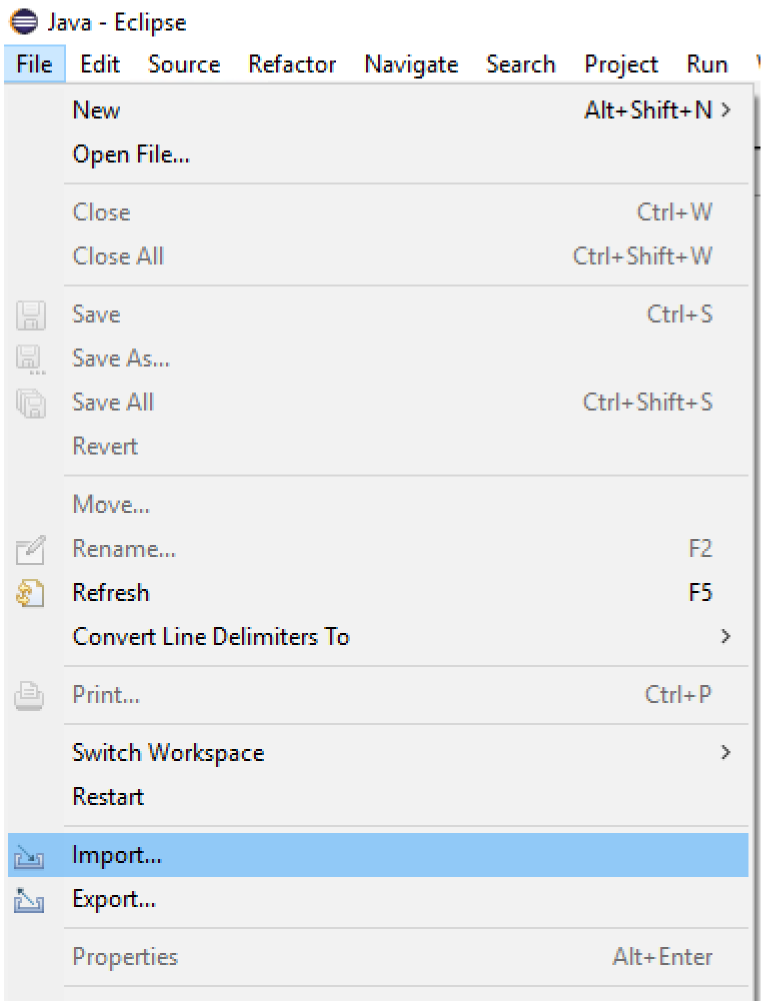
To create a Verification Application project, follow the steps below:

1. Importing the Project from Distribution
2. Configuring the Project
3. Building the Project

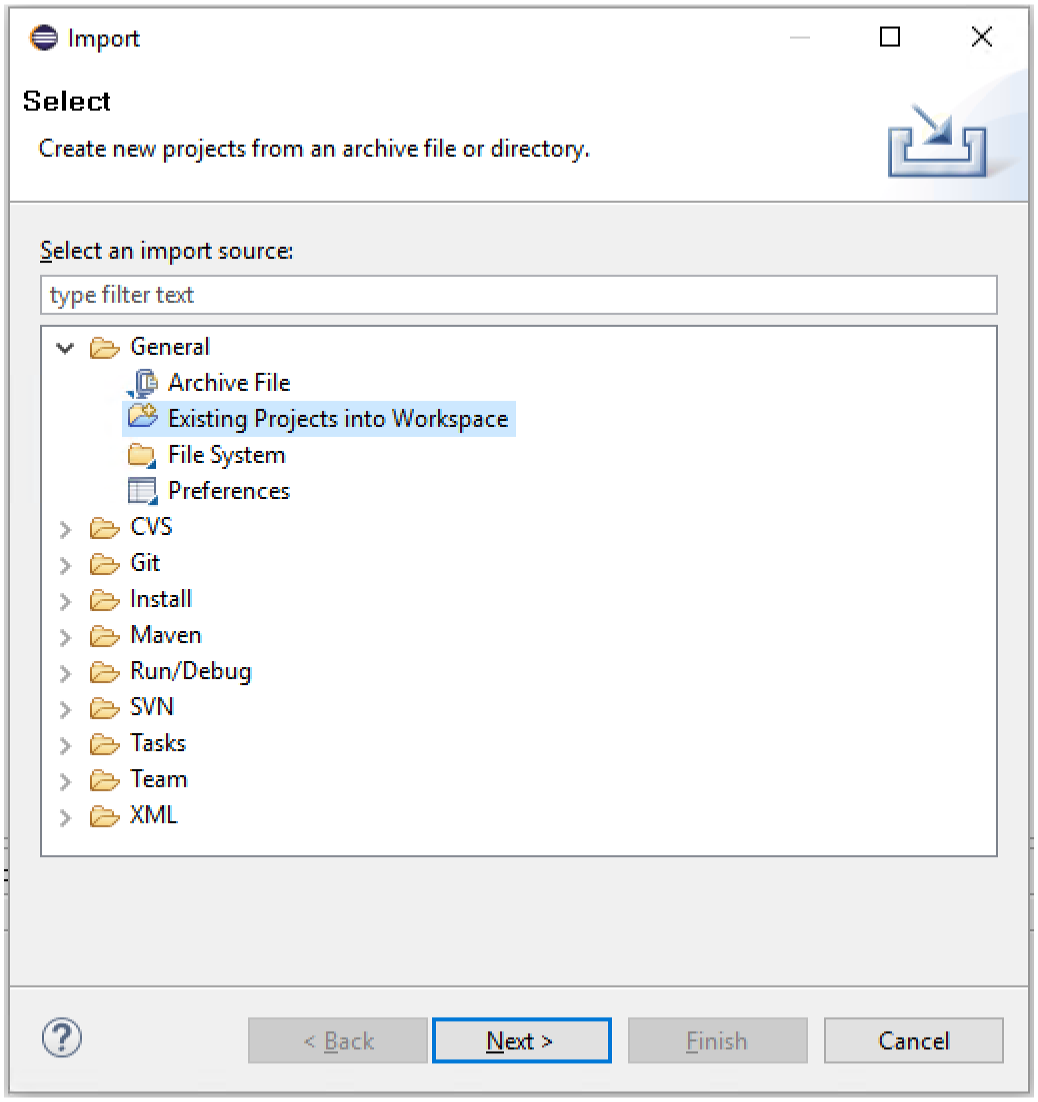
Importing the Project from Distribution

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

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



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

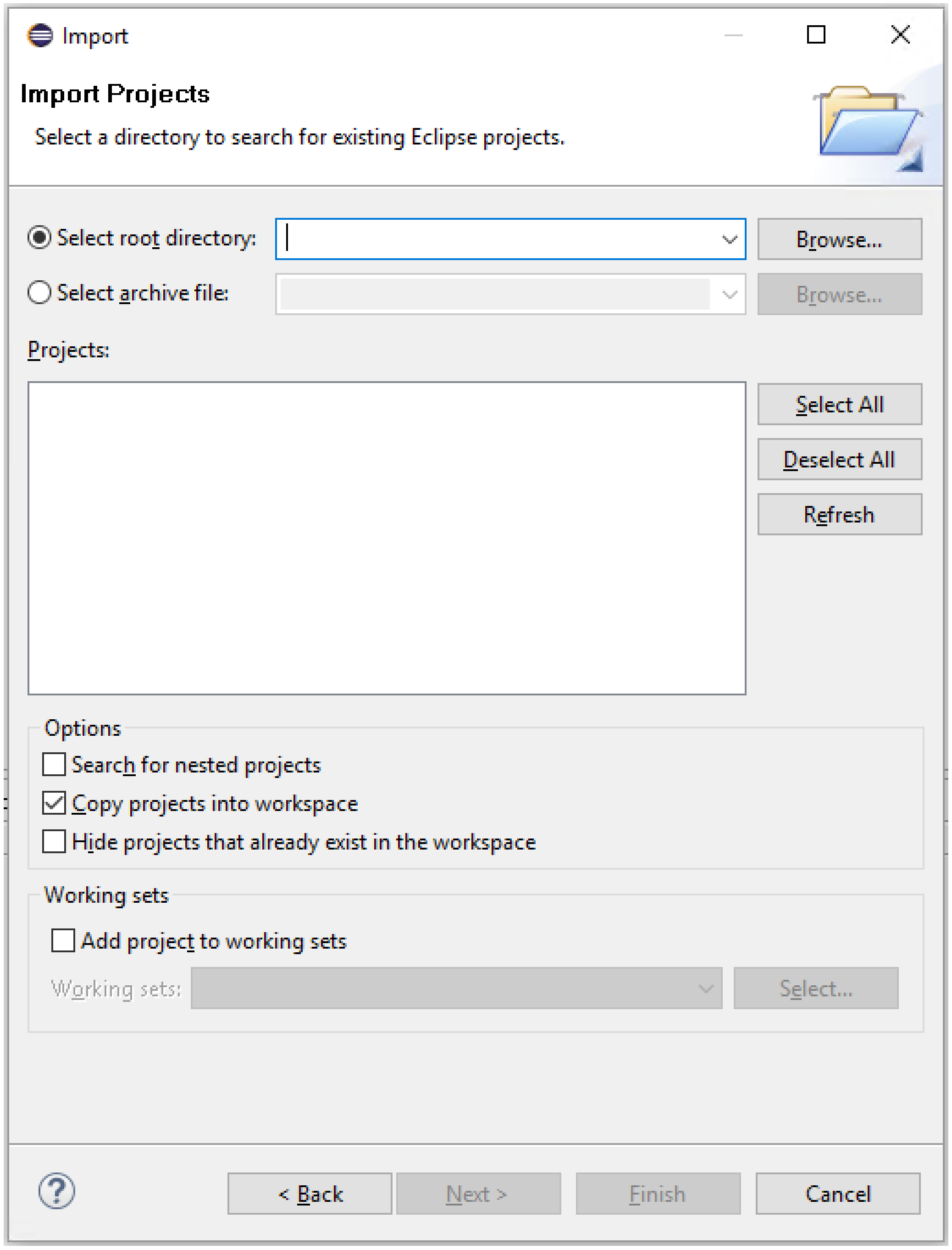


Select

Create new projects from an archive file or directory.

Select an import source:

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



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

☐ Search for nested projects

☑ Copy projects into workspace

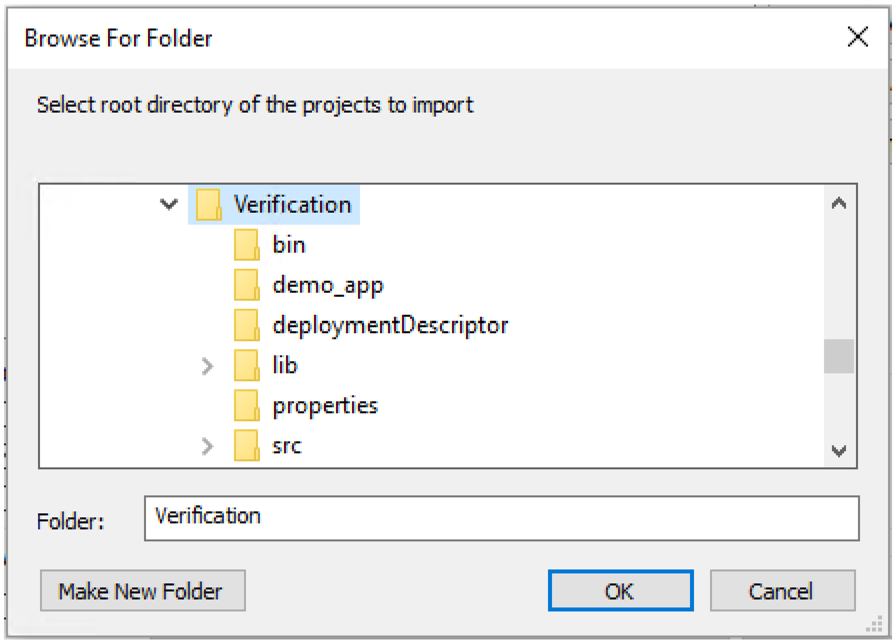
☐ Hide projects that already exist in the workspace

Working sets

☐ Add project to working sets

Working sets:

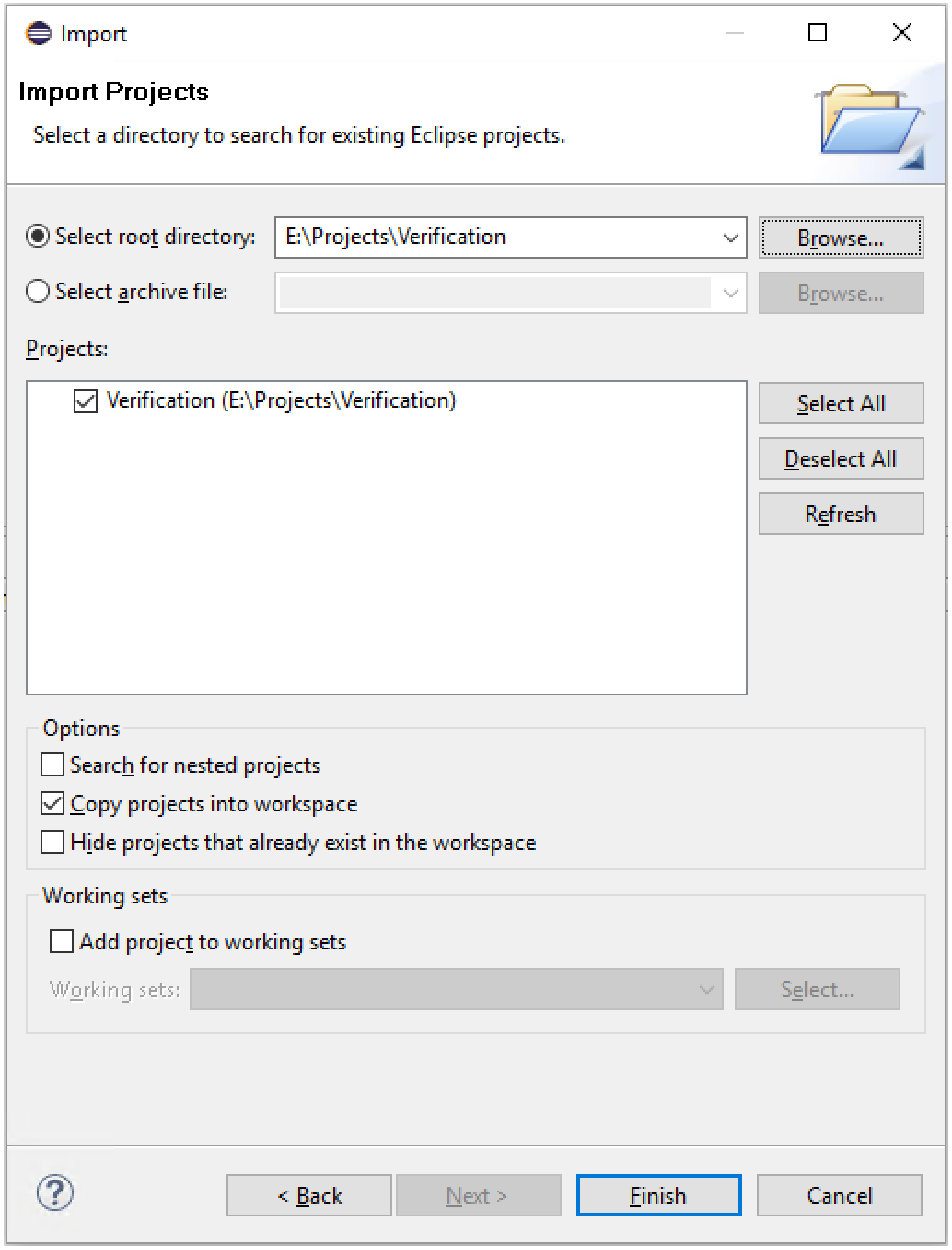
5. In the **Browse For Folder** window, navigate to the extracted project directory and click **Ok.**


Browse For Folder

Select root directory of the projects to import

Folder: Verification

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

[● Select root directory:] E:\Projects\Verification

○ Select archive file:

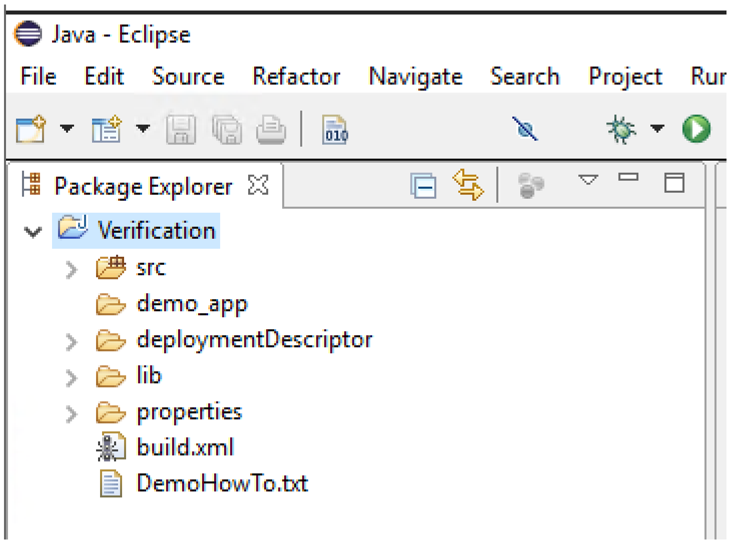
Projects:

- [✓] Verification (E:\Projects\Verification)

Options
- [ ] Search for nested projects
- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets
- [ ] Add project to working sets

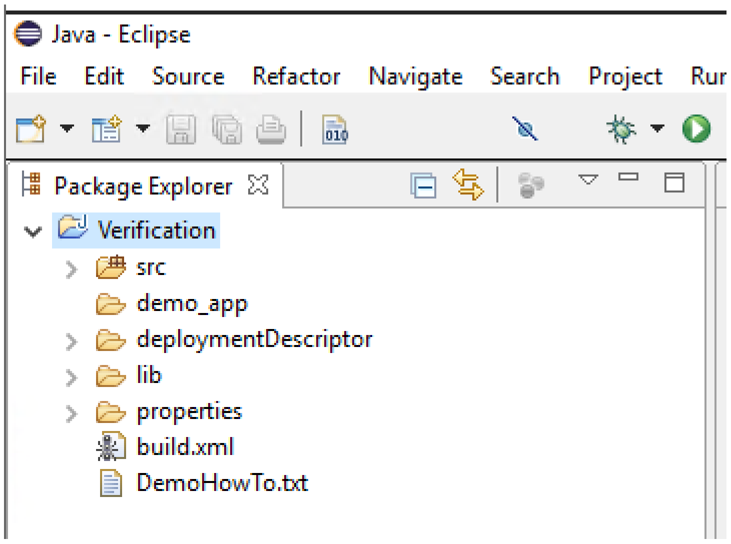
This will now create a project and its content will be shown under **Package Explorer** of the KDE.



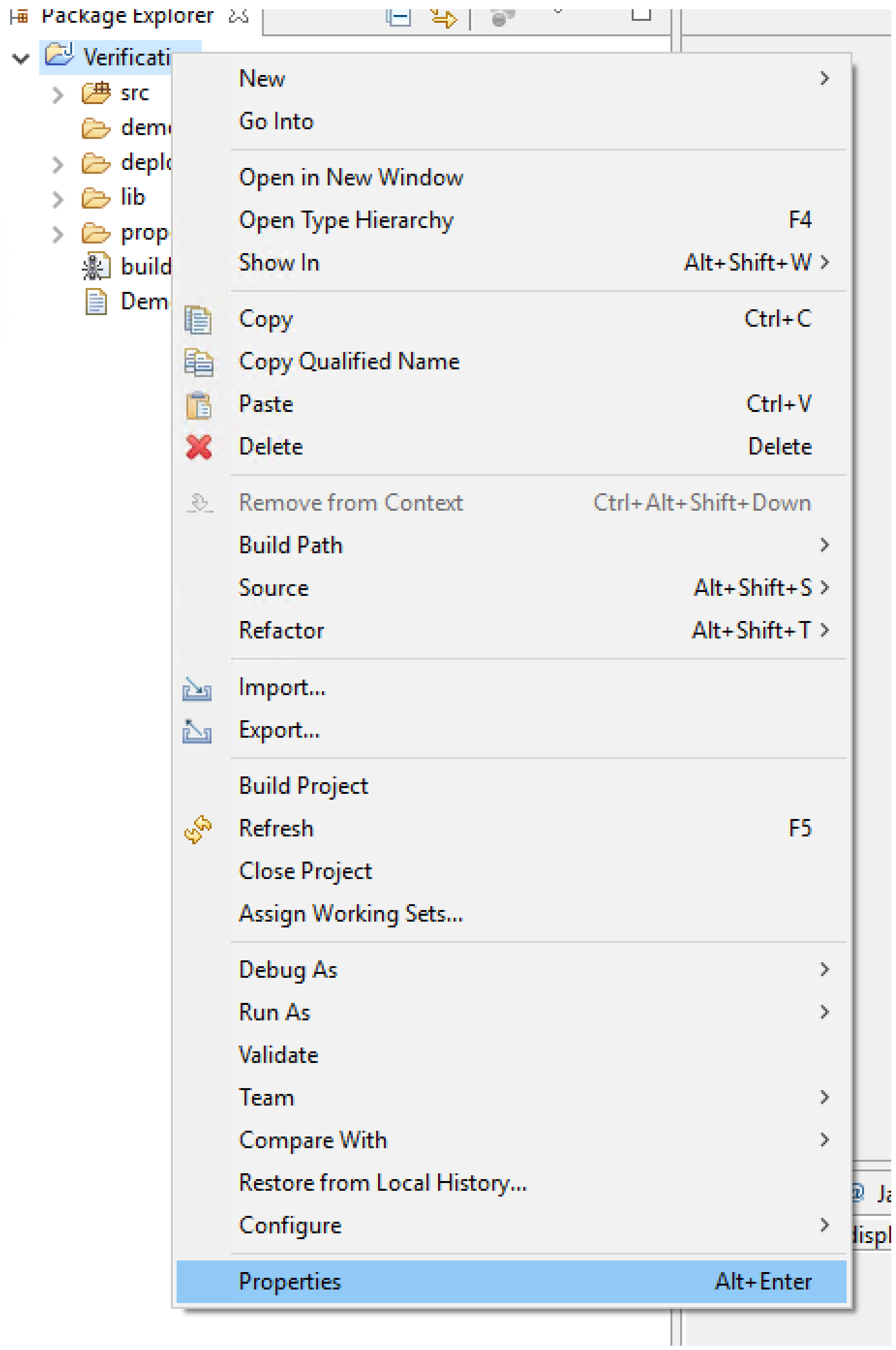
Configuring the Project

Configure the project as follows:

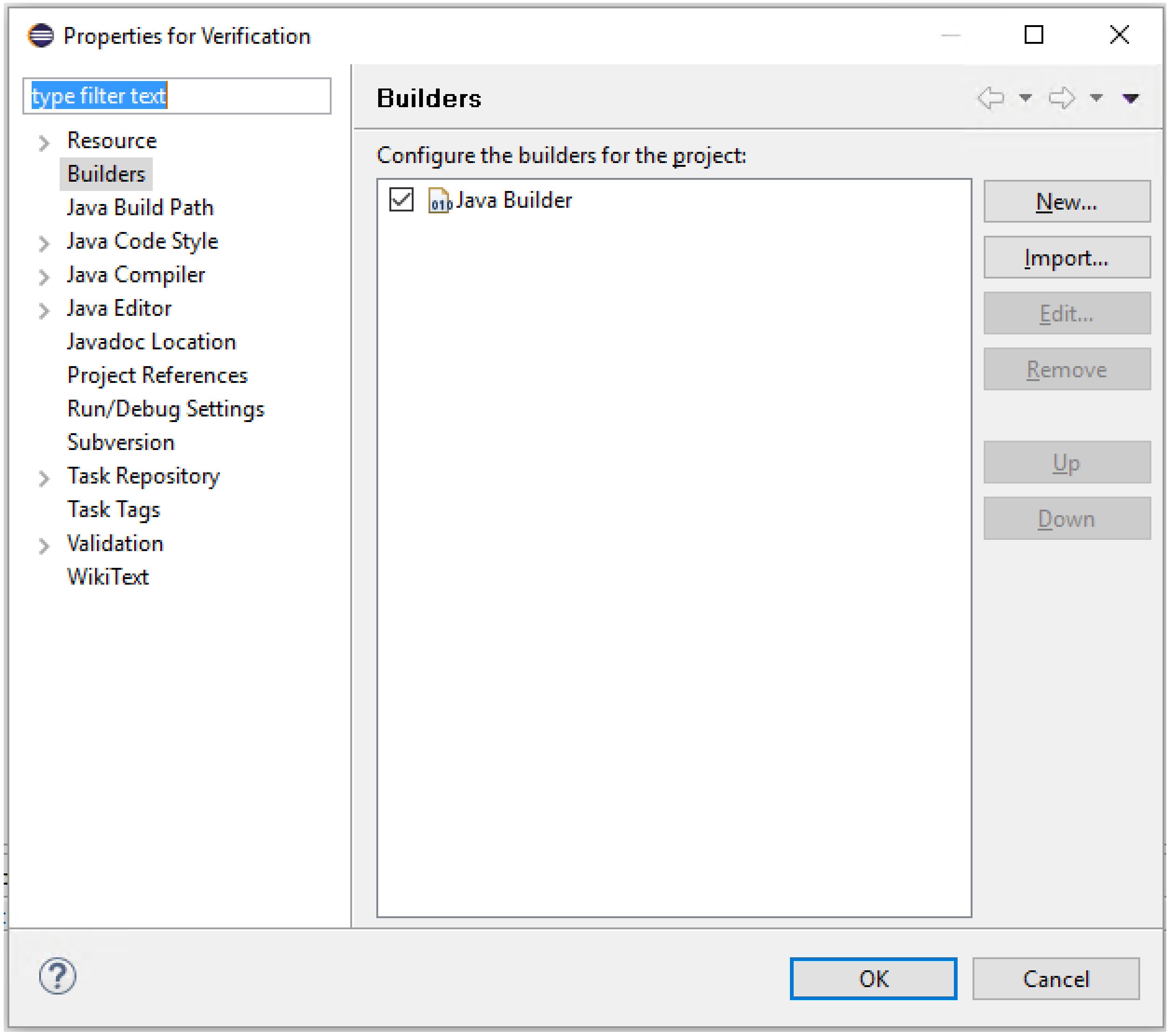
1. Expand the newly imported project in Eclipse as shown below.



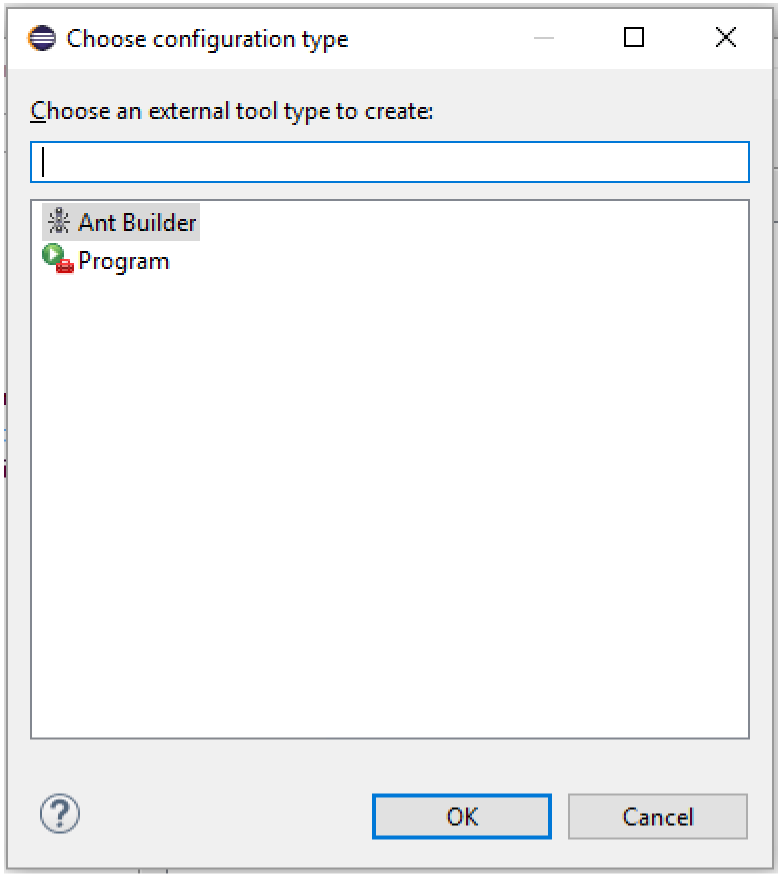
2. Select the project, right click it, and then click **Properties**.



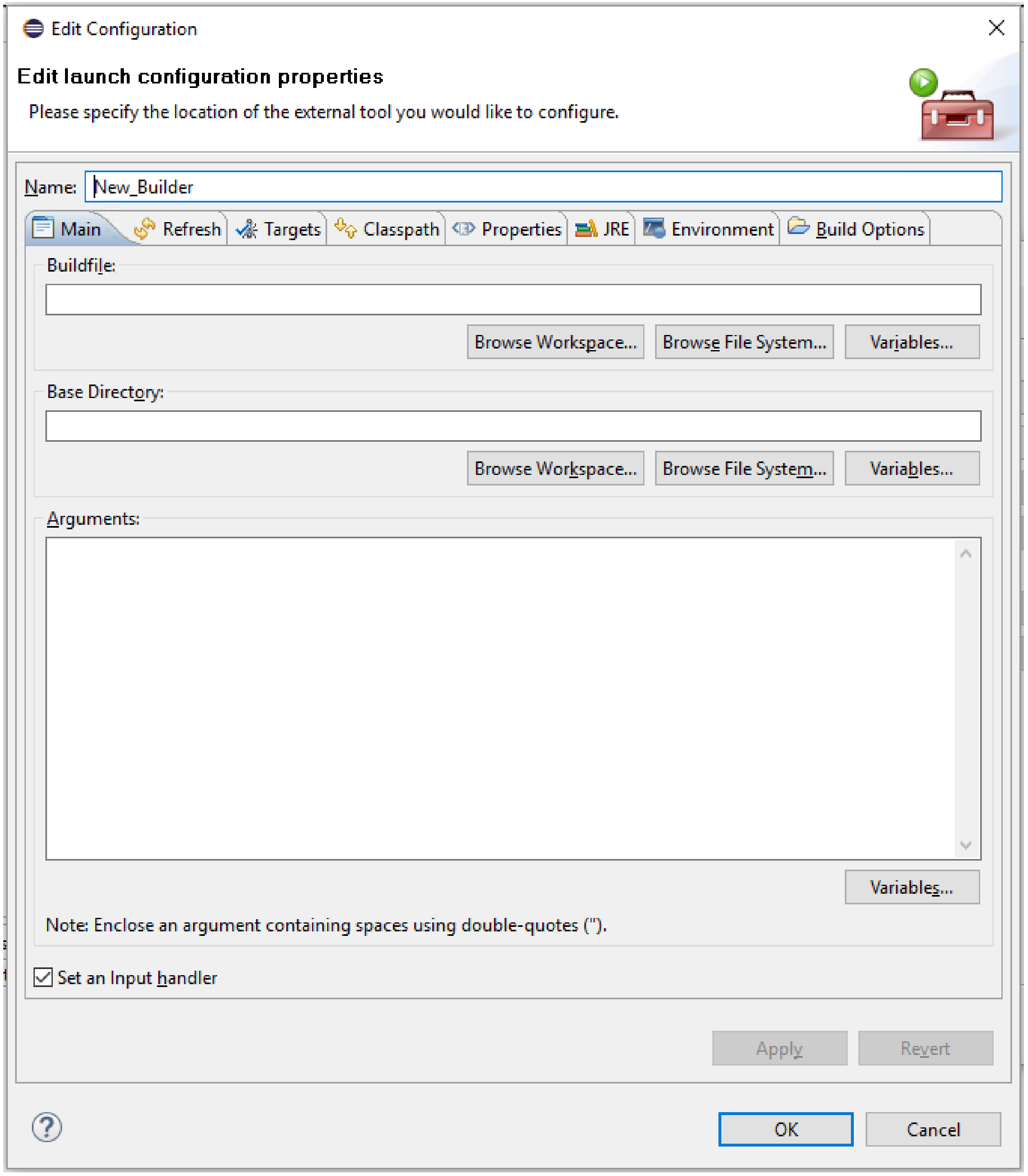
3. In the **Properties for Verification** configuration window, click **Builders** and then click **New**.



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


Choose configuration type
Choose an external tool type to create:
Ant Builder
Program
OK
Cancel

The following **Edit Configuration** window appears.



Edit Configuration

Edit launch configuration properties

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

Name: New_Builder

Buildfile:

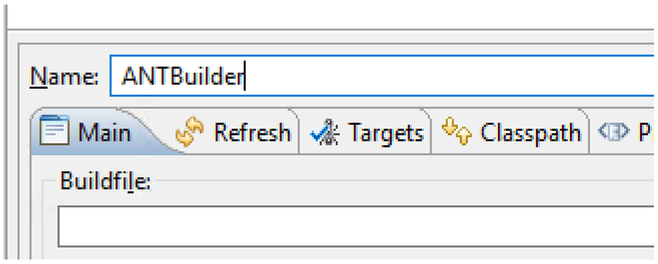
Base Directory:

Arguments:

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

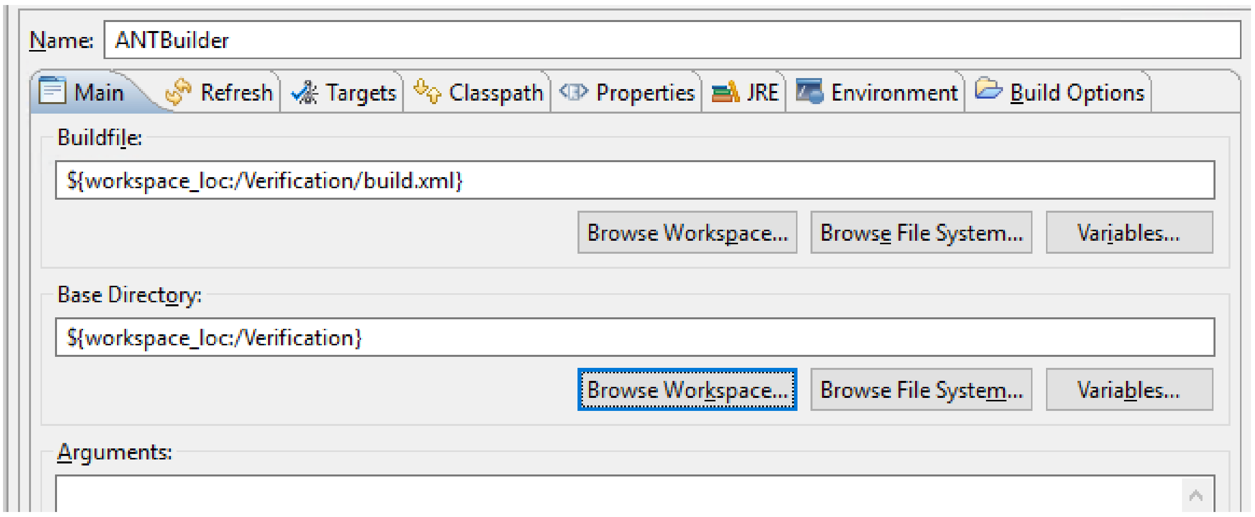
☑ Set an Input handler

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



Name: ANTBuilder

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



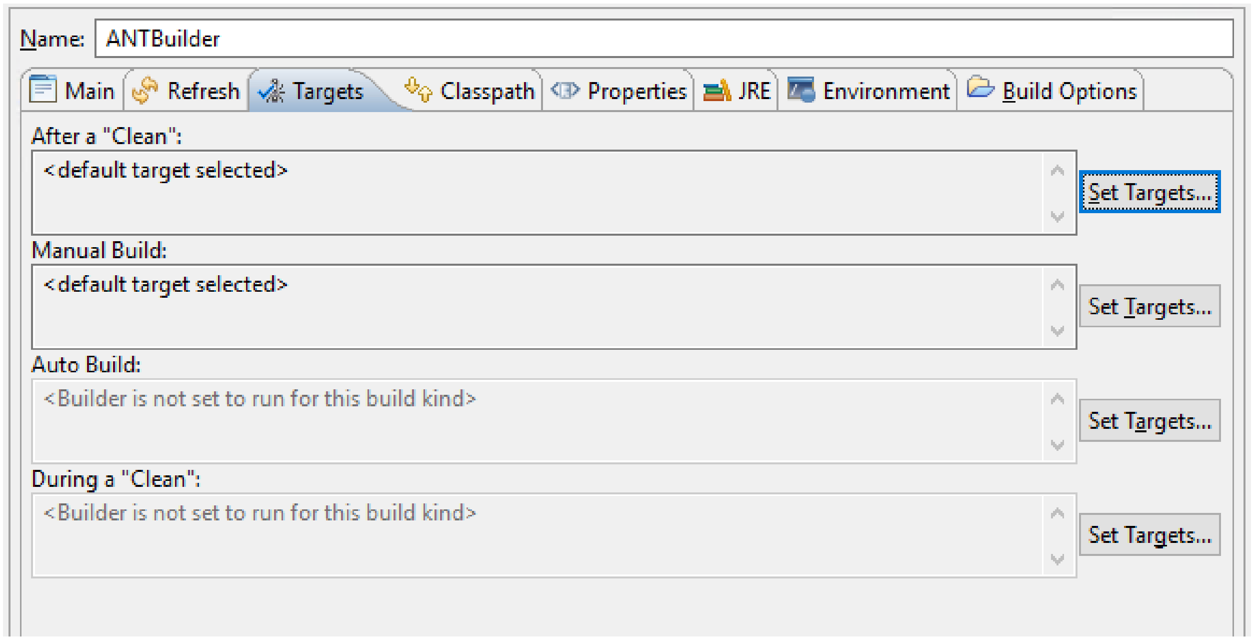
Name: ANTBuilder

Buildfile: ${workspace_loc:/Verification/build.xml}

Base Directory: ${workspace_loc:/Verification}

Arguments:

   c. Click the **Targets** tab.



Name: ANTBuilder

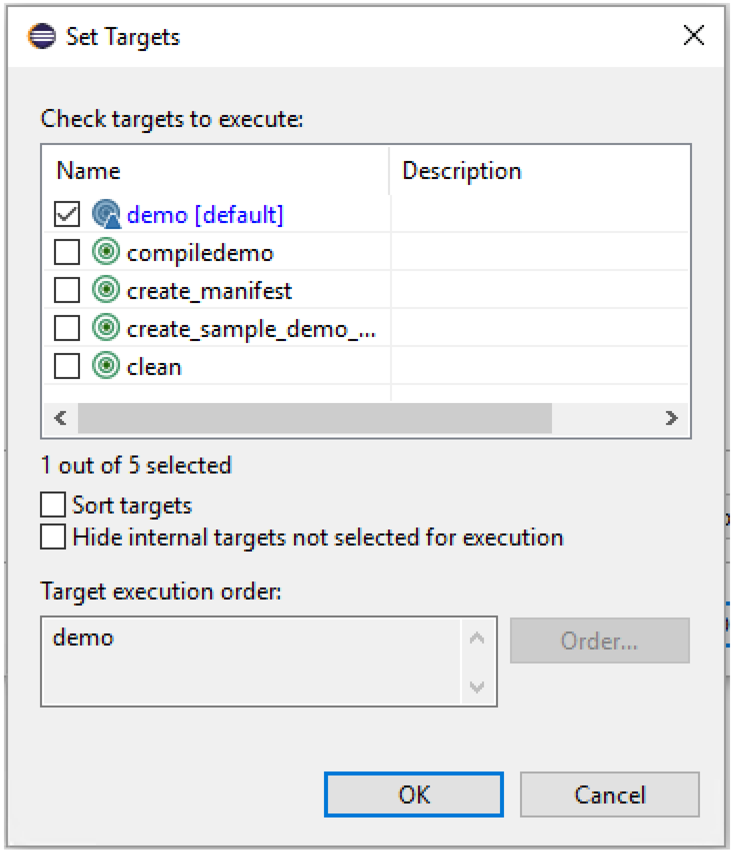
After a "Clean": <default target selected>

Manual Build: <default target selected>

Auto Build: <Builder is not set to run for this build kind>

During a "Clean": <Builder is not set to run for this build kind>

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



Set Targets

Check targets to execute:

Name	Description
✓ demo [default]	
☐ compiledemo	
☐ create_manifest	
☐ create_sample_demo_...	
☐ clean	

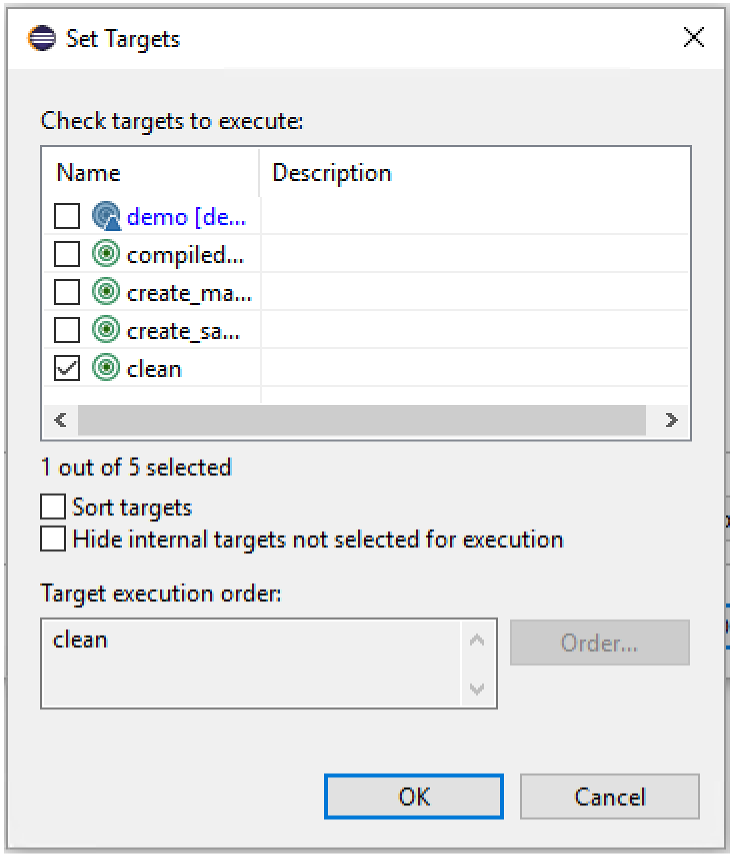
1 out of 5 selected

☐ Sort targets
☐ Hide internal targets not selected for execution

Target execution order:

demo

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


Set Targets

Check targets to execute:

	Name	Description
☐	demo [de...	
☐	compiled...	
☐	create_ma...	
☐	create_sa...	
✓	clean	

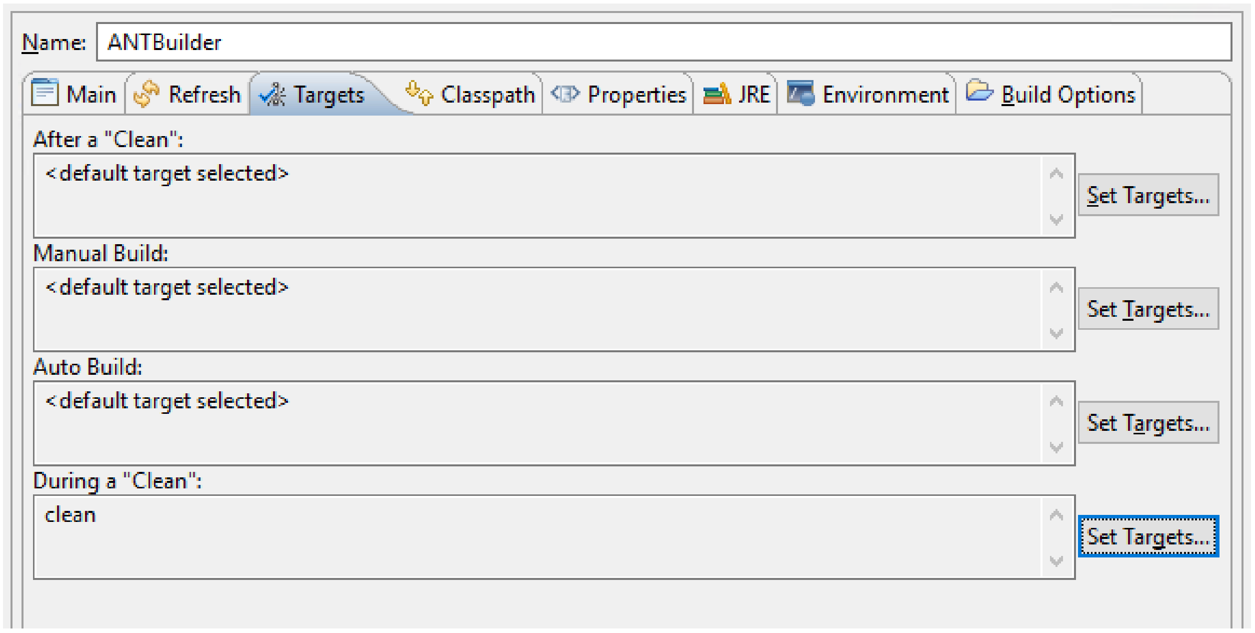
1 out of 5 selected

☐ Sort targets
☐ Hide internal targets not selected for execution

Target execution order:

clean

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


Name: ANTBuilder

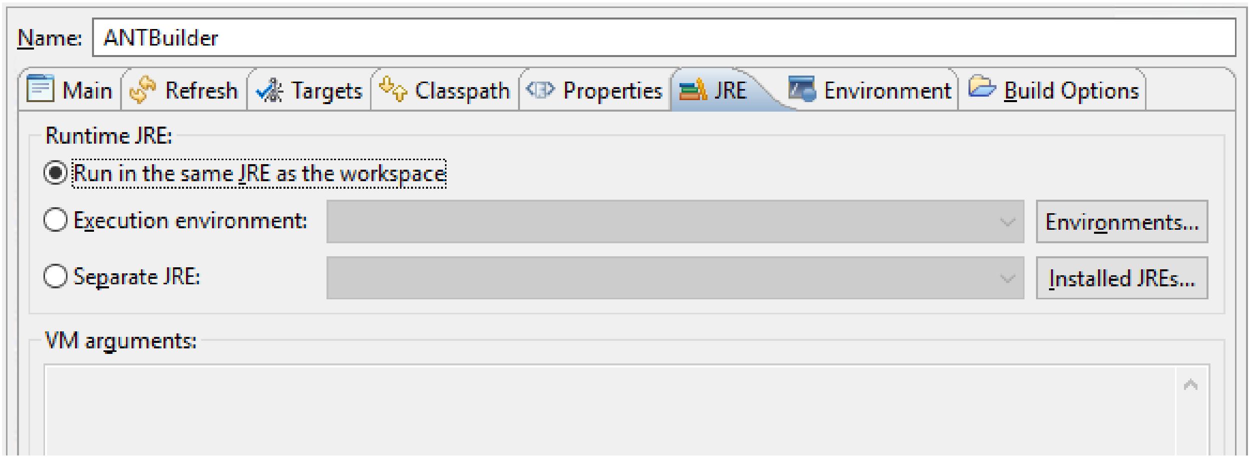
After a "Clean":
<default target selected>

Manual Build:
<default target selected>

Auto Build:
<default target selected>

During a "Clean":
clean

f. Click the **JRE** tab.



Name: ANTBuilder

Runtime JRE:

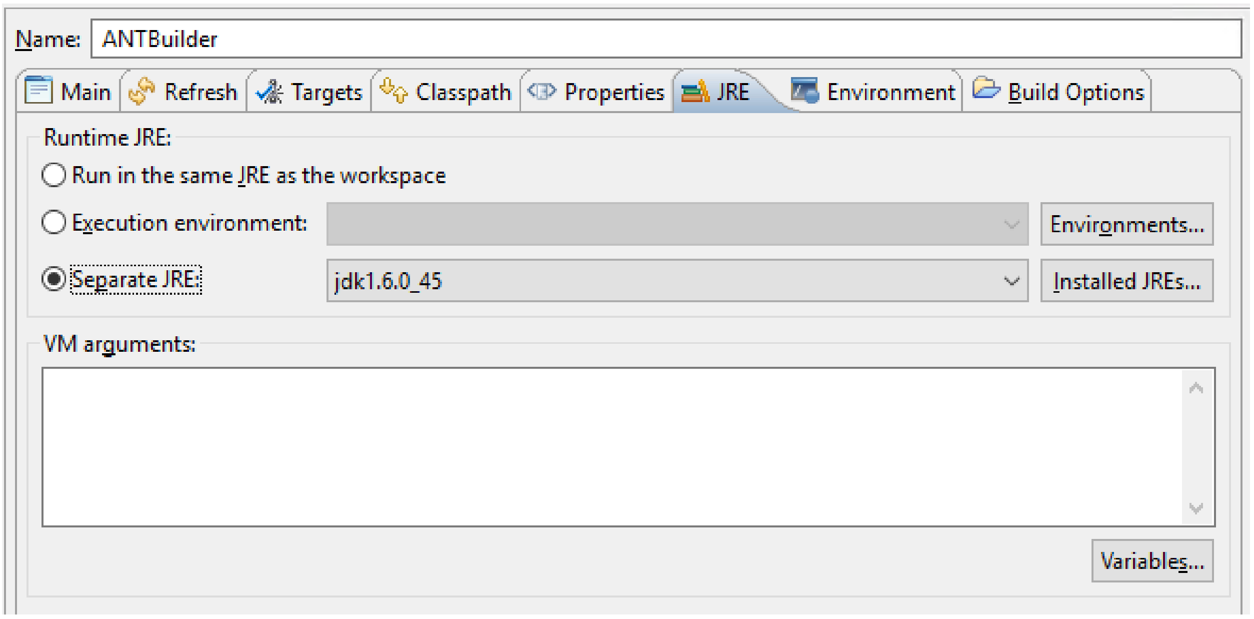
(●) Run in the same JRE as the workspace

( ) Execution environment:

( ) Separate JRE:

VM arguments:

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



Name: ANTBuilder

Runtime JRE:

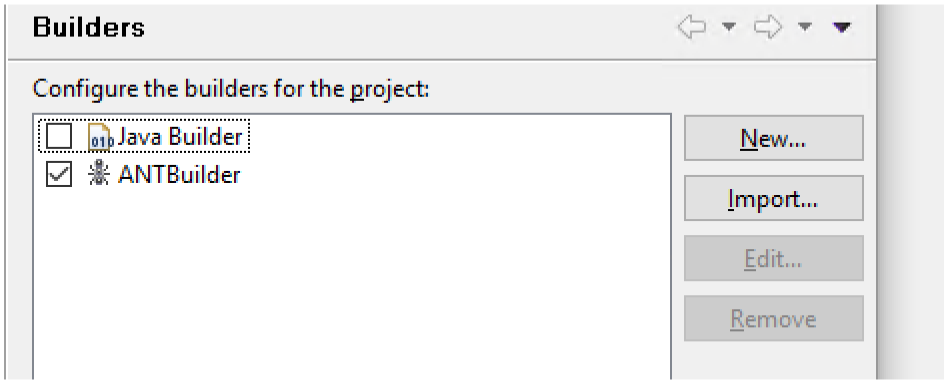
( ) Run in the same JRE as the workspace

( ) Execution environment:

(●) Separate JRE: jdk1.6.0_45

VM arguments:

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



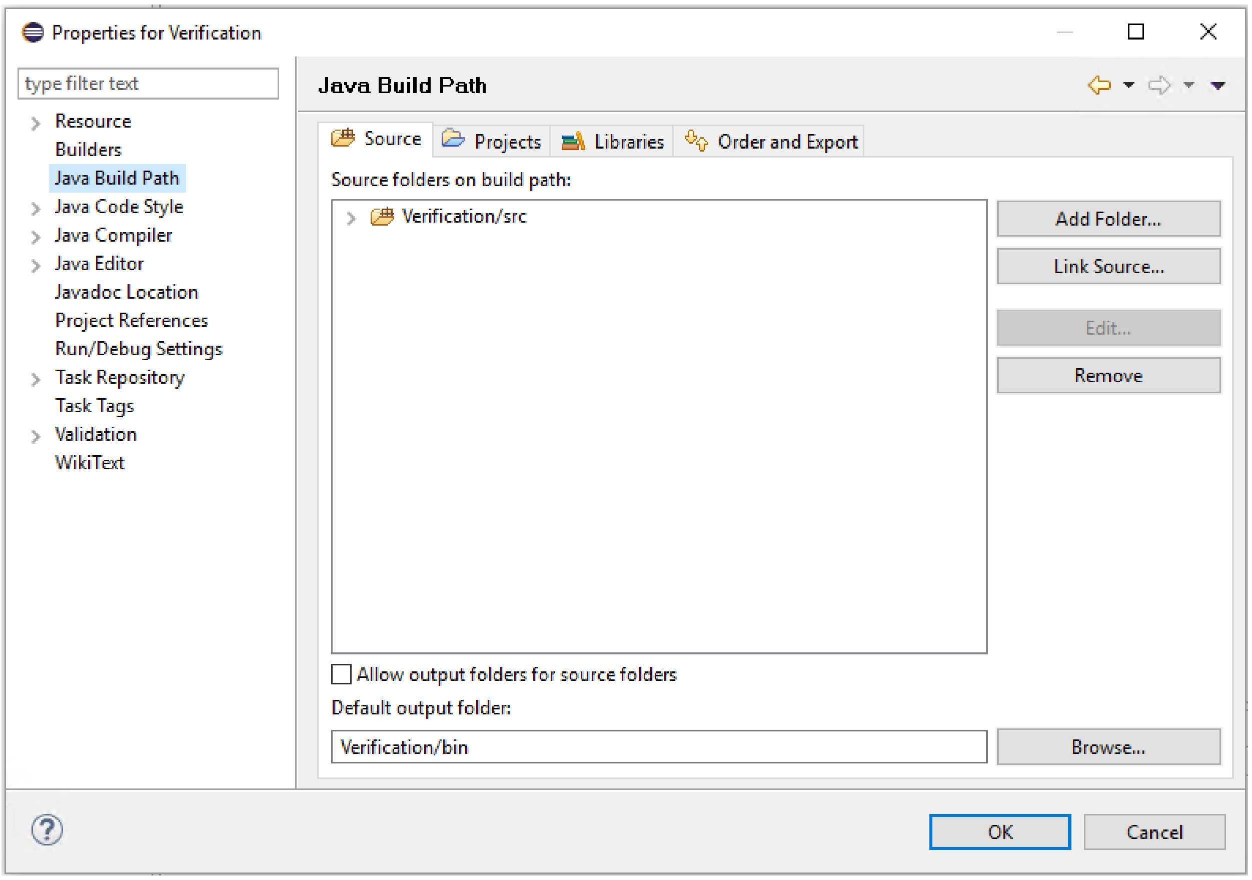
Builders

Configure the builders for the project:

☐ Java Builder

☑ ANTBuilder

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



Source folders on build path:

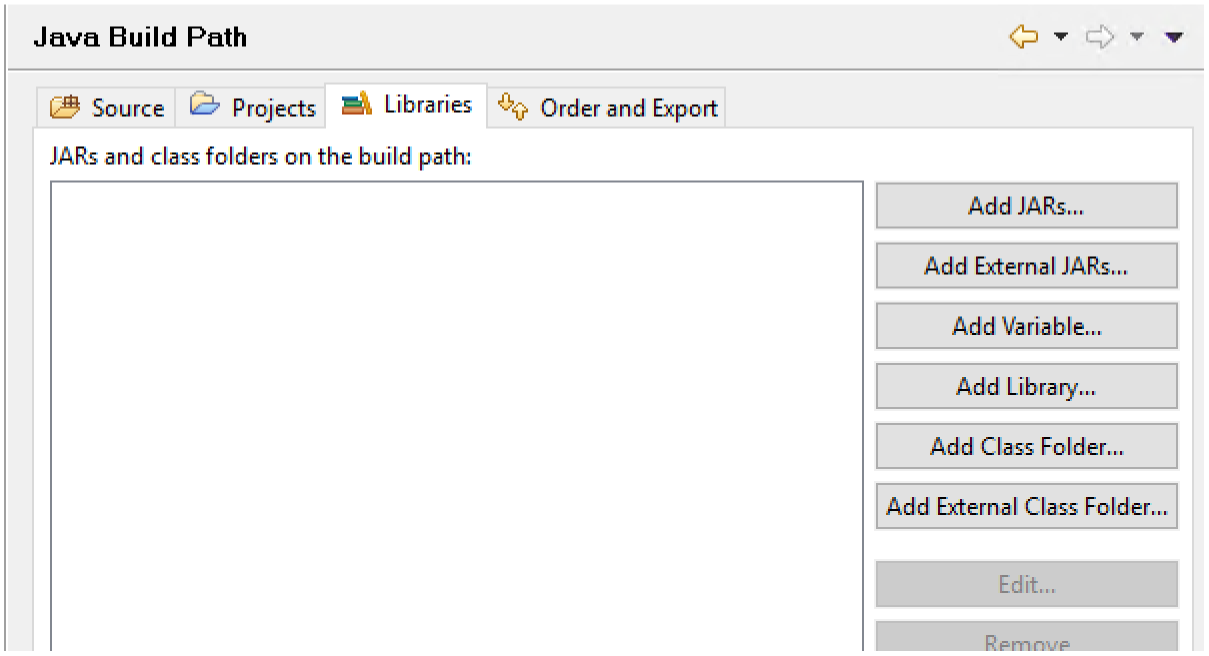
Verification/src

[ ] Allow output folders for source folders

Default output folder:

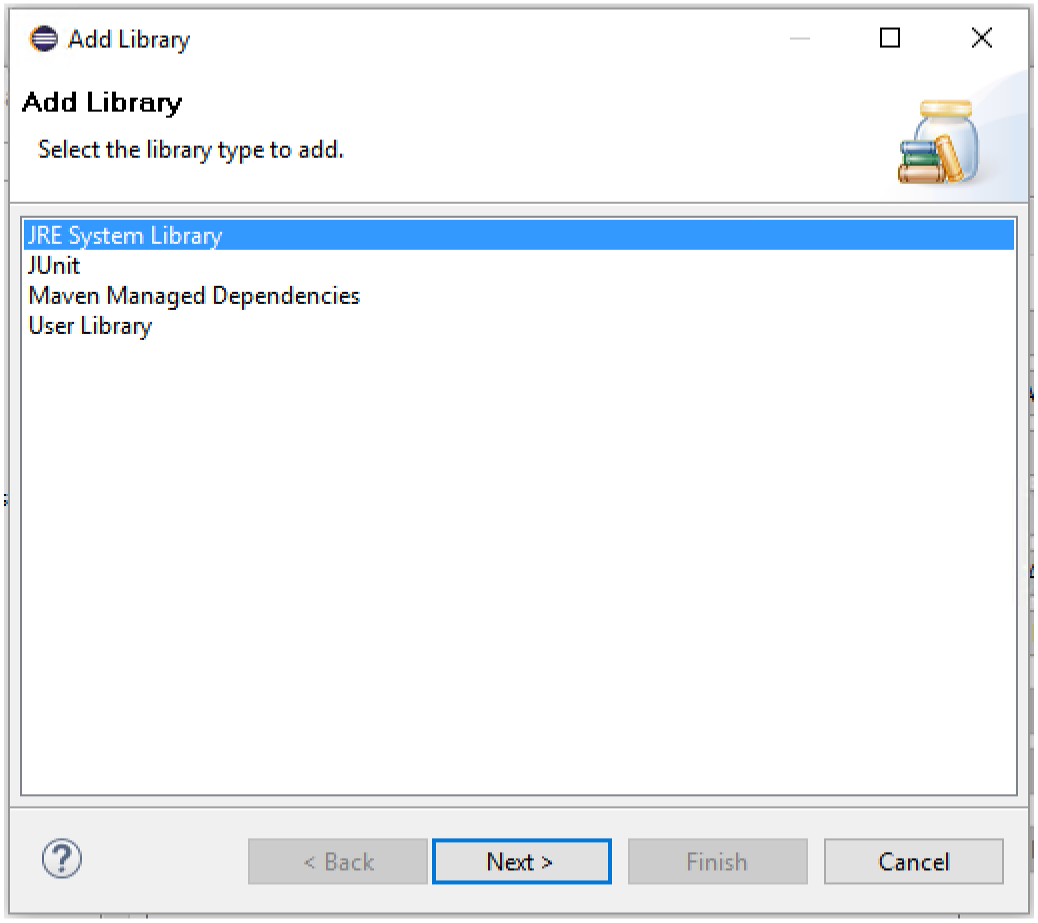
Verification/bin

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



JARs and class folders on the build path:

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



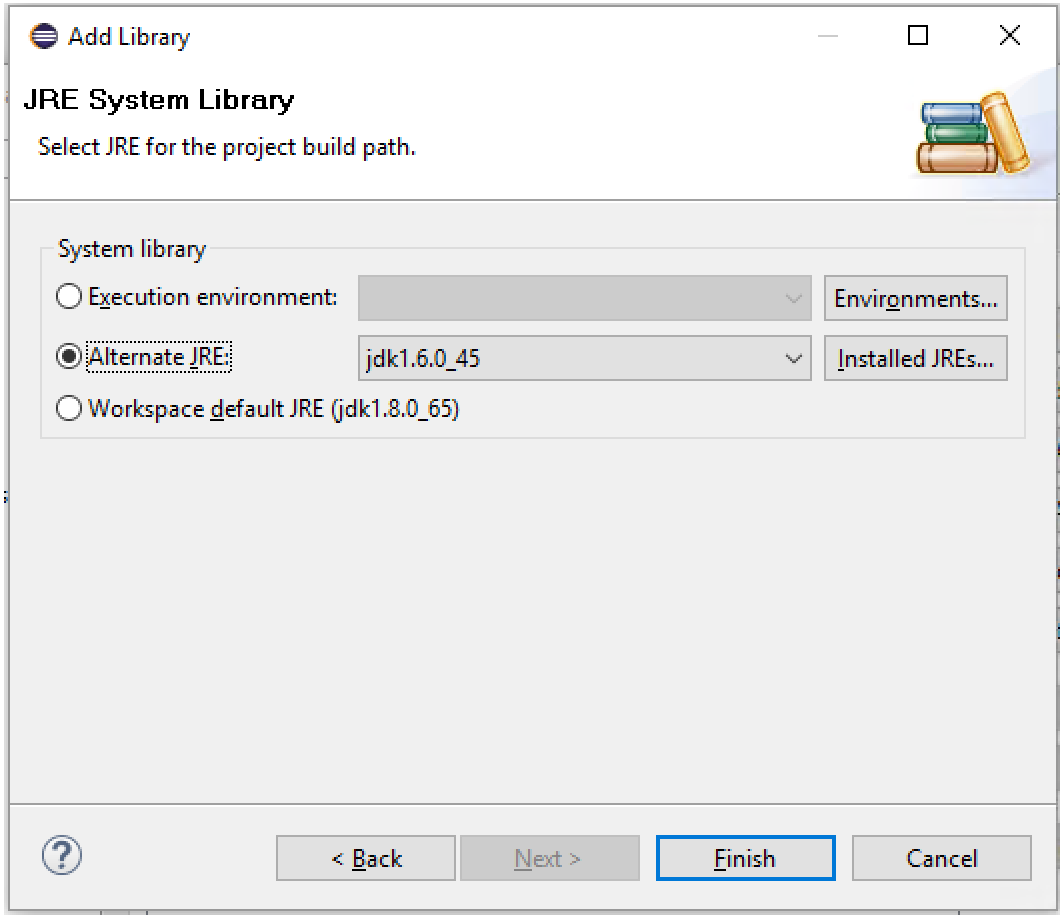
Add Library

Add Library

Select the library type to add.

JRE System Library
JUnit
Maven Managed Dependencies
User Library

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


Add Library

JRE System Library

Select JRE for the project build path.

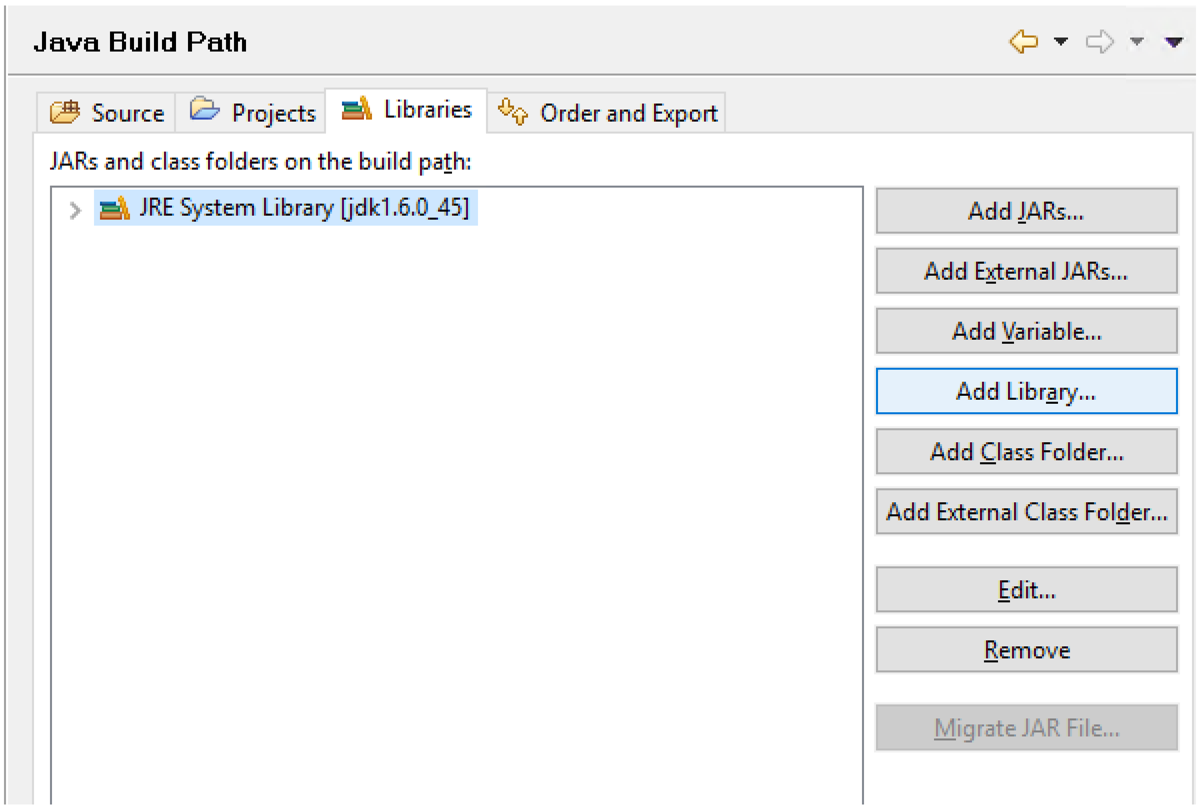
System library

( ) Execution environment: [Environments...]

[● Alternate JRE:] jdk1.6.0_45 [Installed JREs...]

( ) Workspace default JRE (jdk1.8.0_65)

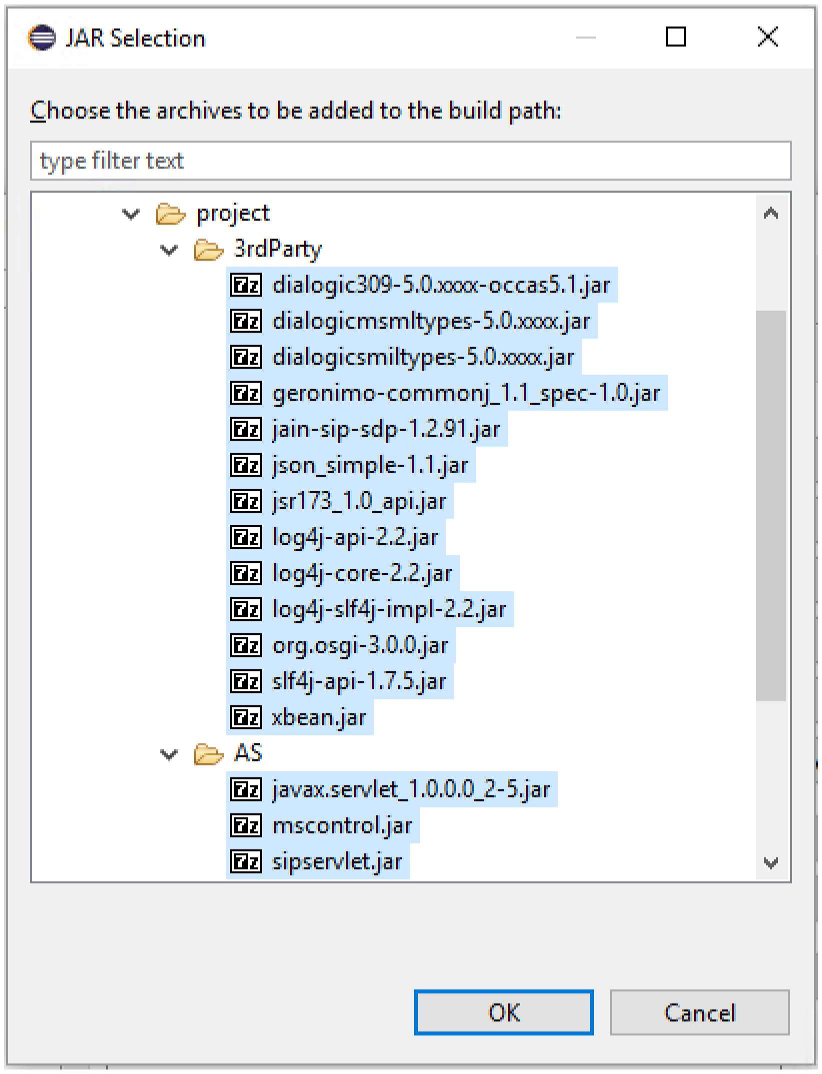
11. To specify project-dependent libraries (JAR) files, click **Add JARs**.



JARs and class folders on the build path:

JRE System Library [jdk1.6.0_45]

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

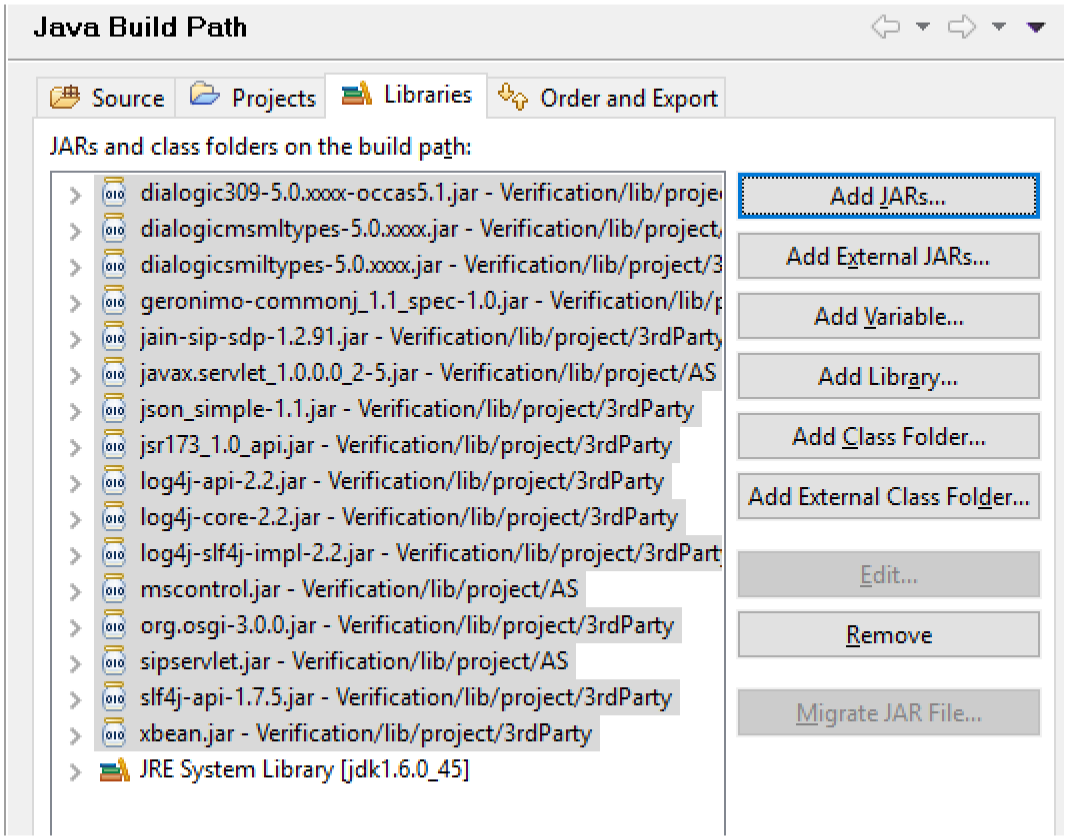


Choose the archives to be added to the build path:

type filter text

- project
  - 3rdParty
    - dialogic309-5.0.xxxx-occas5.1.jar
    - dialogicmsmltypes-5.0.xxxx.jar
    - dialogicsmiltypes-5.0.xxxx.jar
    - geronimo-commonj_1.1_spec-1.0.jar
    - jain-sip-sdp-1.2.91.jar
    - json_simple-1.1.jar
    - jsr173_1.0_api.jar
    - log4j-api-2.2.jar
    - log4j-core-2.2.jar
    - log4j-slf4j-impl-2.2.jar
    - org.osgi-3.0.0.jar
    - slf4j-api-1.7.5.jar
    - xbean.jar
  - AS
    - javax.servlet_1.0.0.0_2-5.jar
    - mscontrol.jar
    - sipservlet.jar

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



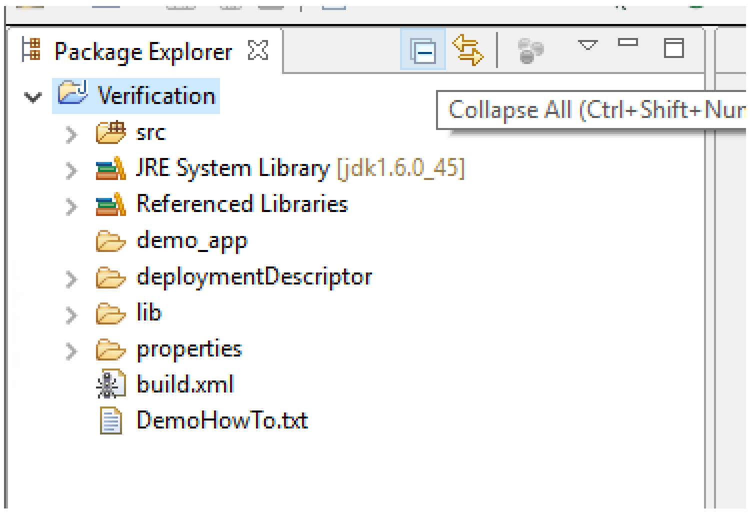
Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- dialogic309-5.0.xxxx-occas5.1.jar - Verification/lib/proje
- dialogicmsmltypes-5.0.xxxx.jar - Verification/lib/project/
- dialogicsmiltypes-5.0.xxxx.jar - Verification/lib/project/3
- geronimo-commonj_1.1_spec-1.0.jar - Verification/lib/p
- jain-sip-sdp-1.2.91.jar - Verification/lib/project/3rdParty
- javax.servlet_1.0.0.0_2-5.jar - Verification/lib/project/AS
- json_simple-1.1.jar - Verification/lib/project/3rdParty
- jsr173_1.0_api.jar - Verification/lib/project/3rdParty
- log4j-api-2.2.jar - Verification/lib/project/3rdParty
- log4j-core-2.2.jar - Verification/lib/project/3rdParty
- log4j-slf4j-impl-2.2.jar - Verification/lib/project/3rdPart
- mscontrol.jar - Verification/lib/project/AS
- org.osgi-3.0.0.jar - Verification/lib/project/3rdParty
- sipservlet.jar - Verification/lib/project/AS
- slf4j-api-1.7.5.jar - Verification/lib/project/3rdParty
- xbean.jar - Verification/lib/project/3rdParty
- JRE System Library [jdk1.6.0_45]

The project is configured and ready to be built.

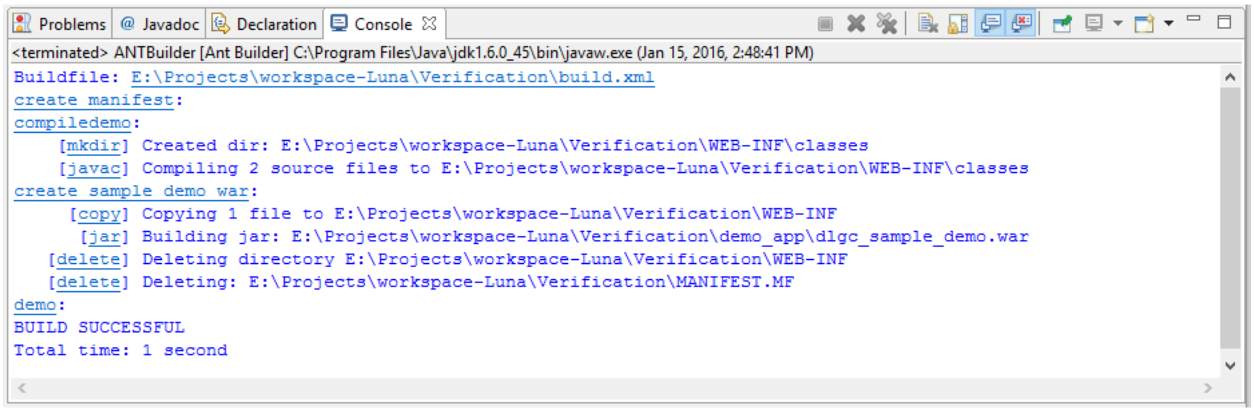


Package Explorer

- Verification
  - src
  - JRE System Library [jdk1.6.0_45]
  - Referenced Libraries
  - demo_app
  - deploymentDescriptor
  - lib
  - properties
  - build.xml
  - DemoHowTo.txt

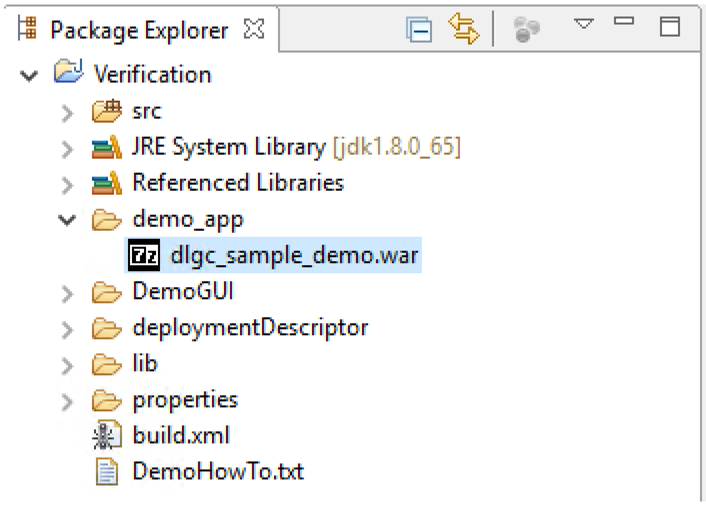
Building the Project

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



```
<terminated> ANTBuilder [Ant Builder] C:\Program Files\Java\jdk1.6.0_45\bin\javaw.exe (Jan 15, 2016, 2:48:41 PM)
Buildfile: E:\Projects\workspace-Luna\Verification\build.xml
create manifest:
compiledemo:
    [mkdir] Created dir: E:\Projects\workspace-Luna\Verification\WEB-INF\classes
    [javac] Compiling 2 source files to E:\Projects\workspace-Luna\Verification\WEB-INF\classes
create sample demo war:
     [copy] Copying 1 file to E:\Projects\workspace-Luna\Verification\WEB-INF
      [jar] Building jar: E:\Projects\workspace-Luna\Verification\demo_app\dlgc_sample_demo.war
   [delete] Deleting directory E:\Projects\workspace-Luna\Verification\WEB-INF
   [delete] Deleting: E:\Projects\workspace-Luna\Verification\MANIFEST.MF
demo:
BUILD SUCCESSFUL
Total time: 1 second
```

The newly built application WAR file will be located in the *DlgcJSR309\application\demo_app* directory named *dlgc_sample_demo.war* file as illustrated below.



Application deployment details can be found in Dialogic JSR 309 Verification Application.

Configuring Eclipse Project and OCCAS Deployed Application for Remote Debugging

In order to connect the newly created project to the deployed WAR file in OCCAS 5.1.0 for debugging purposes, developers need to do the following:

1. Have Eclipse successfully build the Dialogic JSR 309 Verification Application WAR file and deploy it in OCCAS 5.1.0.
2. Configure the OCCAS 5.1.0 for remote debugging:
    a. Stop OCCAS 5.1.0.
    b. In OCCAS 5.1.0, edit the *startWeblogic.sh* script file and add the following line to the exiting "Dialogic additions" section to enable remote debugging as illustrated below.

```
#Dialogic additions
```

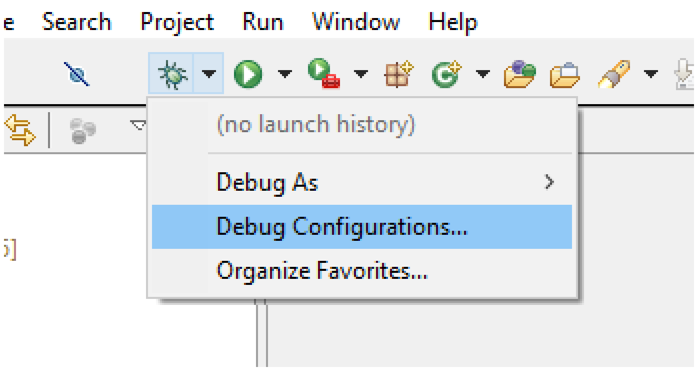
```
export DLG_PROPERTY_FILE=${DOMAIN_HOME}/config/Dialogic/dlgc_JSR309-TCK.properties
LOG4J_OPTIONS="-
Dlog4j.configurationFile=${DOMAIN_HOME}/config/Dialogic/log4j2.xml"
CLASSPATH="${CLASSPATH}:${ORCL_HOME}/server/modules/mscontrol.jar:${ORCL_HOME}/ser
ver/lib/jsr309-descriptor-binding.jar:${DOMAIN_HOME}/lib/org.osgi-
3.0.0.jar:${DOMAIN_HOME}/lib/log4j-api-2.2.jar:${DOMAIN_HOME}/lib/log4j-slf4j-
impl-2.2.jar:${DOMAIN_HOME}/lib/log4j-core-2.2.jar:${DOMAIN_HOME}/lib/slf4j-api-
1.7.5.jar"
SERIALIZATION="-Dwlss.local.serialization=false"

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

DLGC_OPTS="${DEBUG_OPTS} ${SERIALIZATION} ${LOG4J_OPTIONS}"

# END Dialogic additions
```

Note: The socket address specified above is 8787 but any port of choice can be used. Any port used needs to be enabled in a firewall in order to allow communication through it.

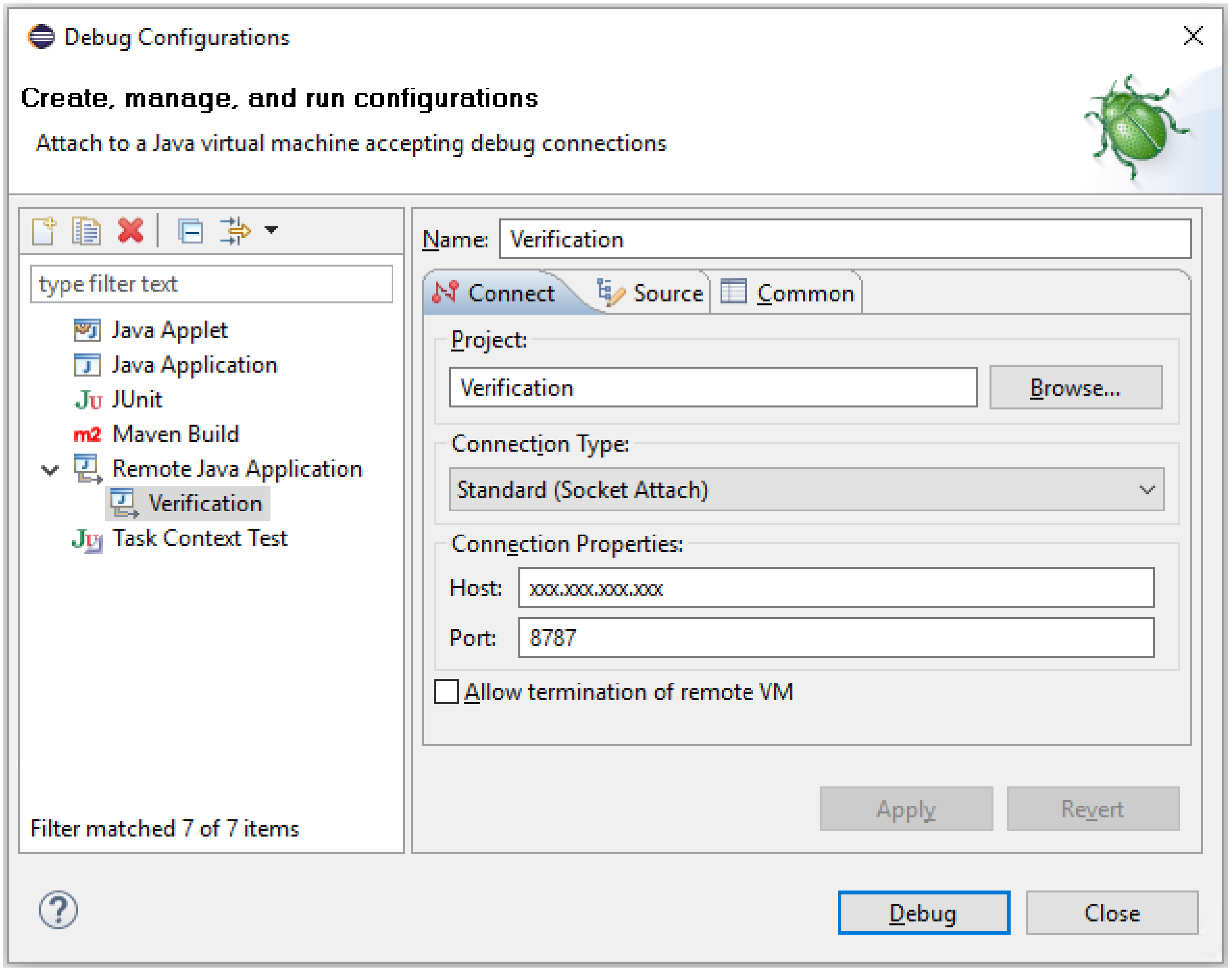
3. Start OCCAS 5.1.0 and make sure there are no errors in the console.

Eclipse Project Remote Debugging Configuration

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



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


Debug Configurations

Create, manage, and run configurations

Attach to a Java virtual machine accepting debug connections

type filter text

- Java Applet
- Java Application
- JUnit
- Maven Build
- Remote Java Application
  - Verification
- Task Context Test

Filter matched 7 of 7 items

Name: Verification

Connect | Source | Common

Project: Verification

Connection Type: Standard (Socket Attach)

Connection Properties:

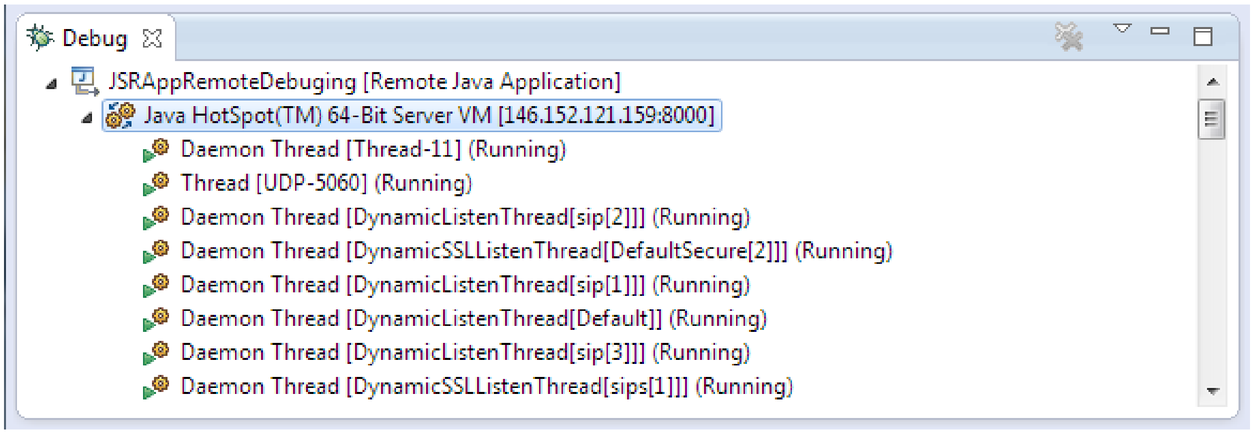
Host: xxx.xxx.xxx.xxx

Port: 8787

☐ Allow termination of remote VM

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

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



```
JSRAppRemoteDebuging [Remote Java Application]
  Java HotSpot(TM) 64-Bit Server VM [146.152.121.159:8000]
    Daemon Thread [Thread-11] (Running)
    Thread [UDP-5060] (Running)
    Daemon Thread [DynamicListenThread[sip[2]]] (Running)
    Daemon Thread [DynamicSSLListenThread[DefaultSecure[2]]] (Running)
    Daemon Thread [DynamicListenThread[sip[1]]] (Running)
    Daemon Thread [DynamicListenThread[Default]] (Running)
    Daemon Thread [DynamicListenThread[sip[3]]] (Running)
    Daemon Thread [DynamicSSLListenThread[sips[1]]] (Running)
```

Now, the Eclipse project is connected to the build application that is deployed in OCCAS 5.1.

7. Appendix A: JSR 309 Connector Environment Setup

About

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

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

This section does not go into the details of OCCAS 5.1, but it will help build an OCCAS 5.1 system that can be used for verification purposes.

Note: It is recommended to setup your system with NTP for single source of time so each system in the architecture can have the same time reference: Enable NTP (Network Time Protocol).

If you need more details on OCCAS 5.1, refer to the OCCAS 5.1 installation instructions available from www.oracle.com.

Installing and Configuring OCCAS

This section describes how to install and configure OCCAS 5.1 in order to be able to proceed to Installing the Dialogic JSR 309 Connector.

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

Note: If you are familiar with OCCAS or are planning to deploy on an existing OCCAS 5.1 setup, proceed to Installing the Dialogic JSR 309 Connector.

Preinstallation Setup

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

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

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

2. Run the following command at the prompt:

   ```
   service network restart
   ```

3. Create a non-root user account that OCCAS 5.1 will be installed under.

   ```
   useradd occas5.1
   ```

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

   ```
   passwd occas5.1
   ```

4. Download the latest JDK .rpm file from www.oracle.com and install the JDK .rpm file. For example, the following setup is based on *jdk-6u45-linux-x64.rpm* file.

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

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

```
# .bashrc

export JAVA_HOME=/usr/java/jdk1.6.0_45

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

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

```
source ./bashrc
```

OCCAS Installation

Refer to Oracle's OCCAS 5.1 Installation Guide for any further details and additional installation options. Note that the GUI installation does require a graphical capability of the system ("X Window System") in order for the installer to be able to display the GUI.

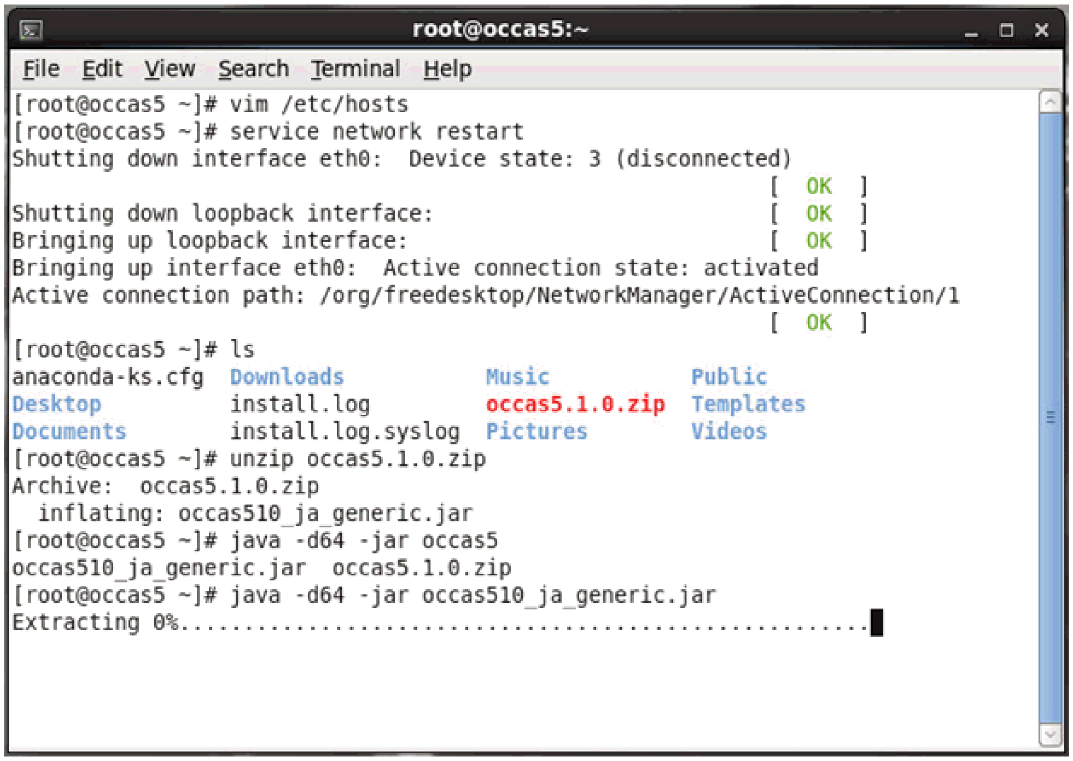
Log in as a non-root user (for example, occas5.1) and place the OCCAS 5.1 installation file (*occas_generic.jar*) in the non-root user home directory (for example, /home/occas5.1).

```
source ./bashrc
```

1. To install OCCAS5.1, execute the following command:

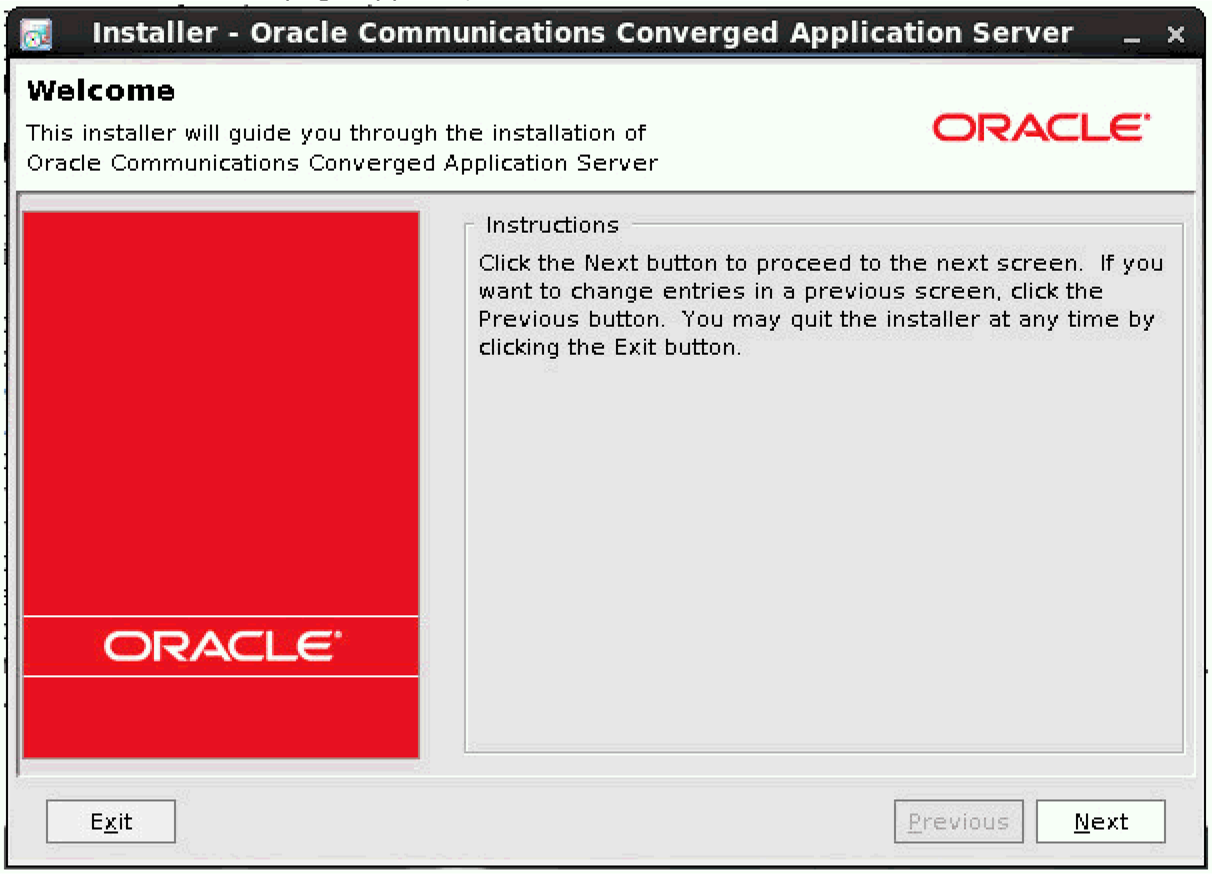
```
java -d64 -jar occas510_ja_generic.jar
```

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



```
[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%....................................................
```

2. When the **Welcome** page appears, click **Next**.



Installer - Oracle Communications Converged Application Server

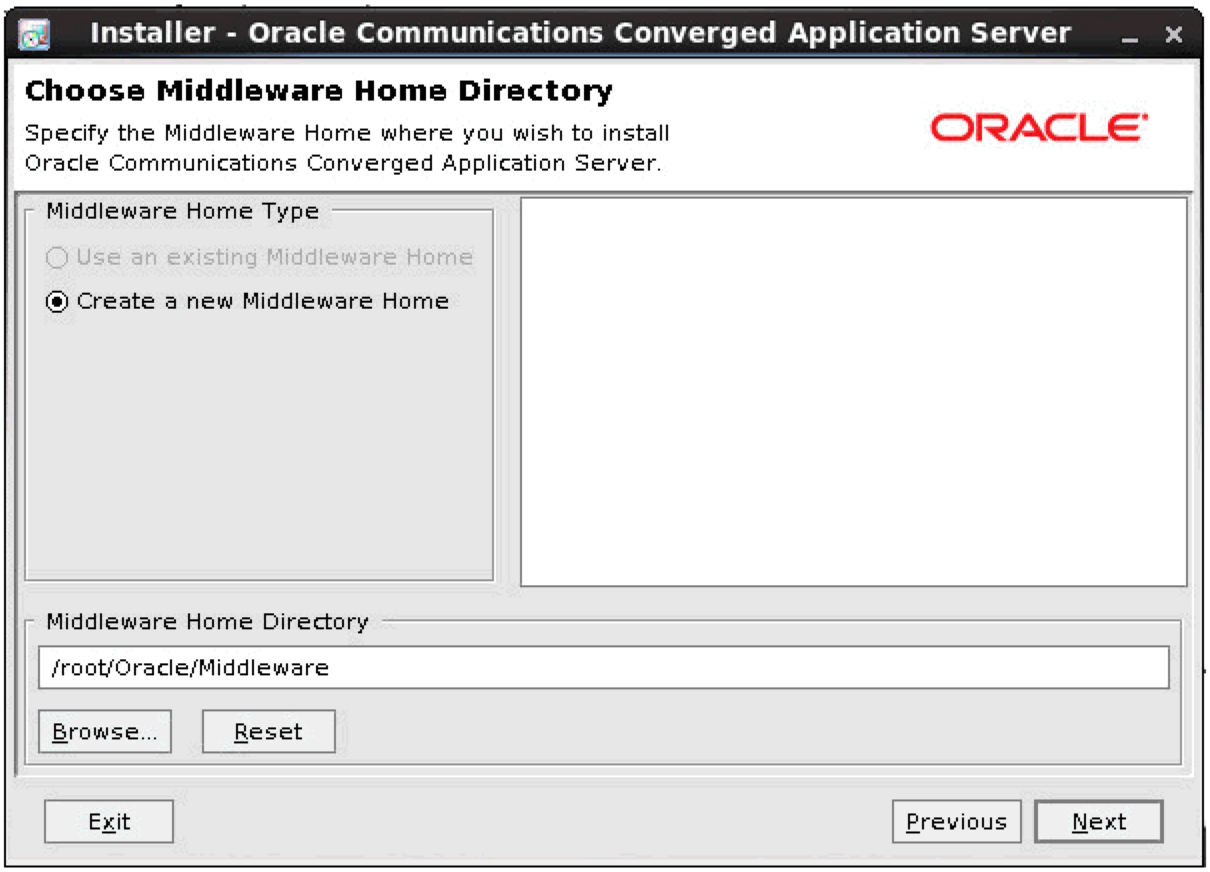
Welcome

This installer will guide you through the installation of Oracle Communications Converged Application Server

Instructions

Click the Next button to proceed to the next screen. If you want to change entries in a previous screen, click the Previous button. You may quit the installer at any time by clicking the Exit button.

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



Installer - Oracle Communications Converged Application Server

Choose Middleware Home Directory

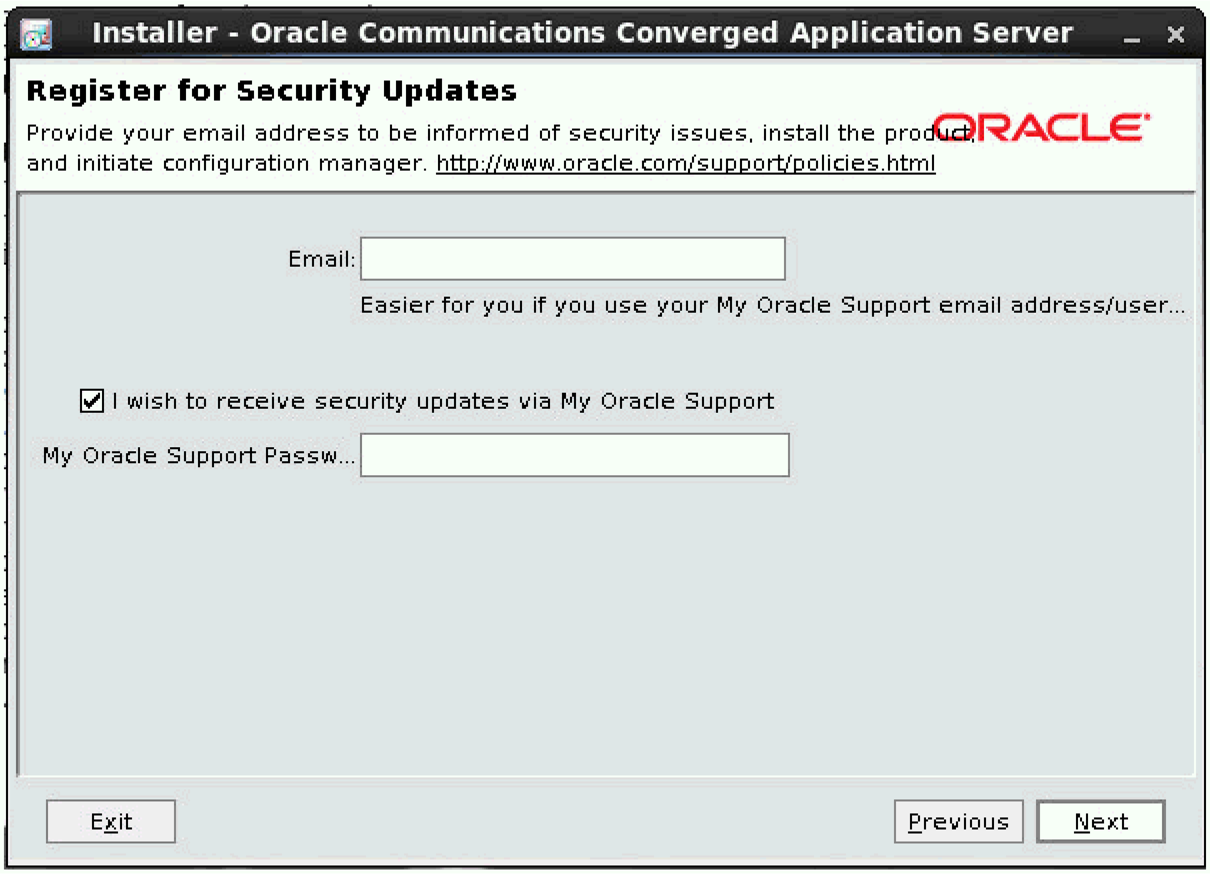
Specify the Middleware Home where you wish to install Oracle Communications Converged Application Server.

Middleware Home Type

Use an existing Middleware Home [✓ Create a new Middleware Home]

Middleware Home Directory: /root/Oracle/Middleware

4. Deselect **I wish to receive security updates via My Oracle Support** (unless you have an account with Oracle), and then click **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

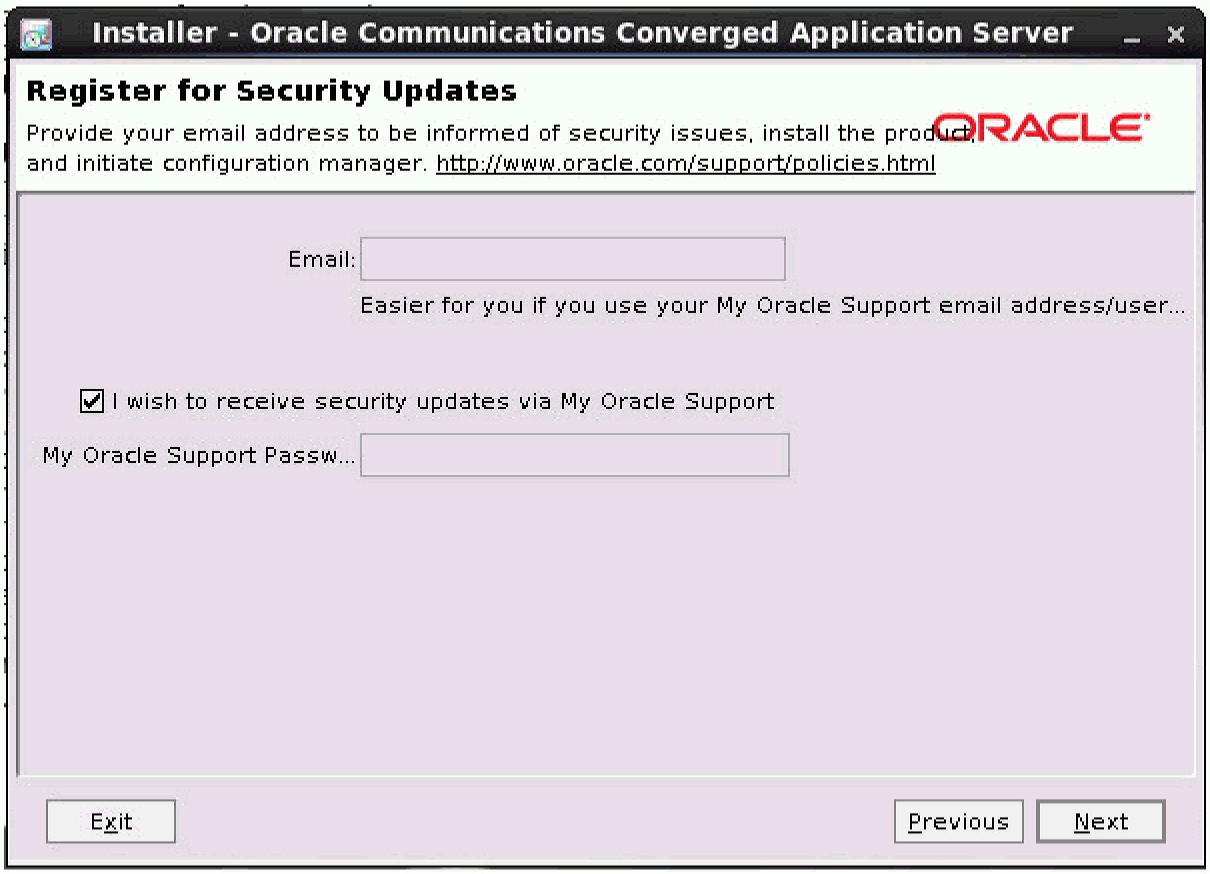
Email:

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...

5. Click **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

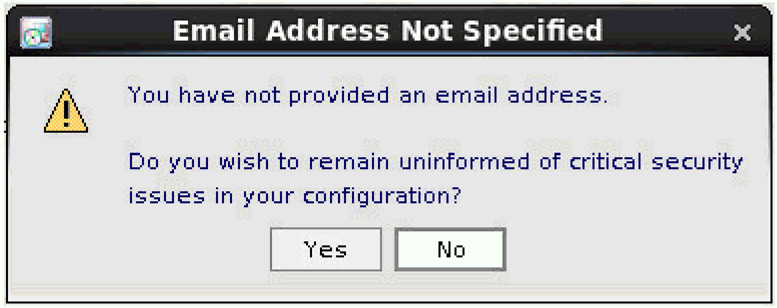
Email:

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...

6. Click **Yes**.

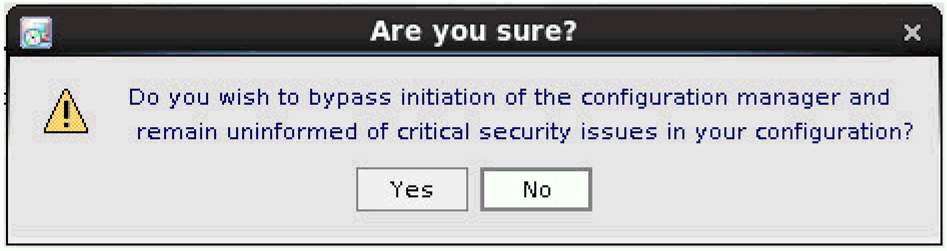


Email Address Not Specified

You have not provided an email address.

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

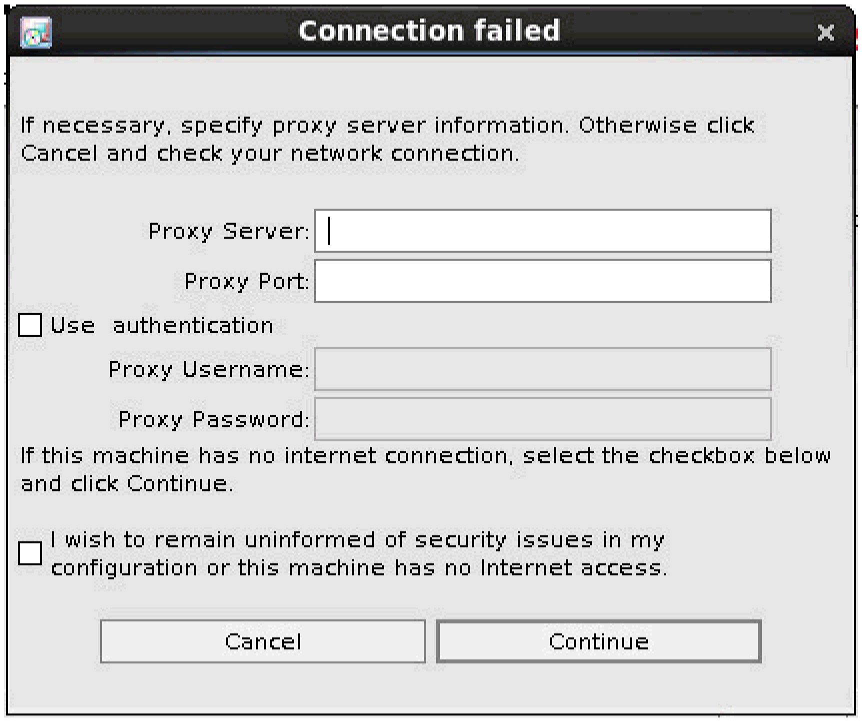
7. Click **Yes**.



Are you sure?

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

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



Connection failed

If necessary, specify proxy server information. Otherwise click Cancel and check your network connection.

Proxy Server:

Proxy Port:

☐ Use authentication

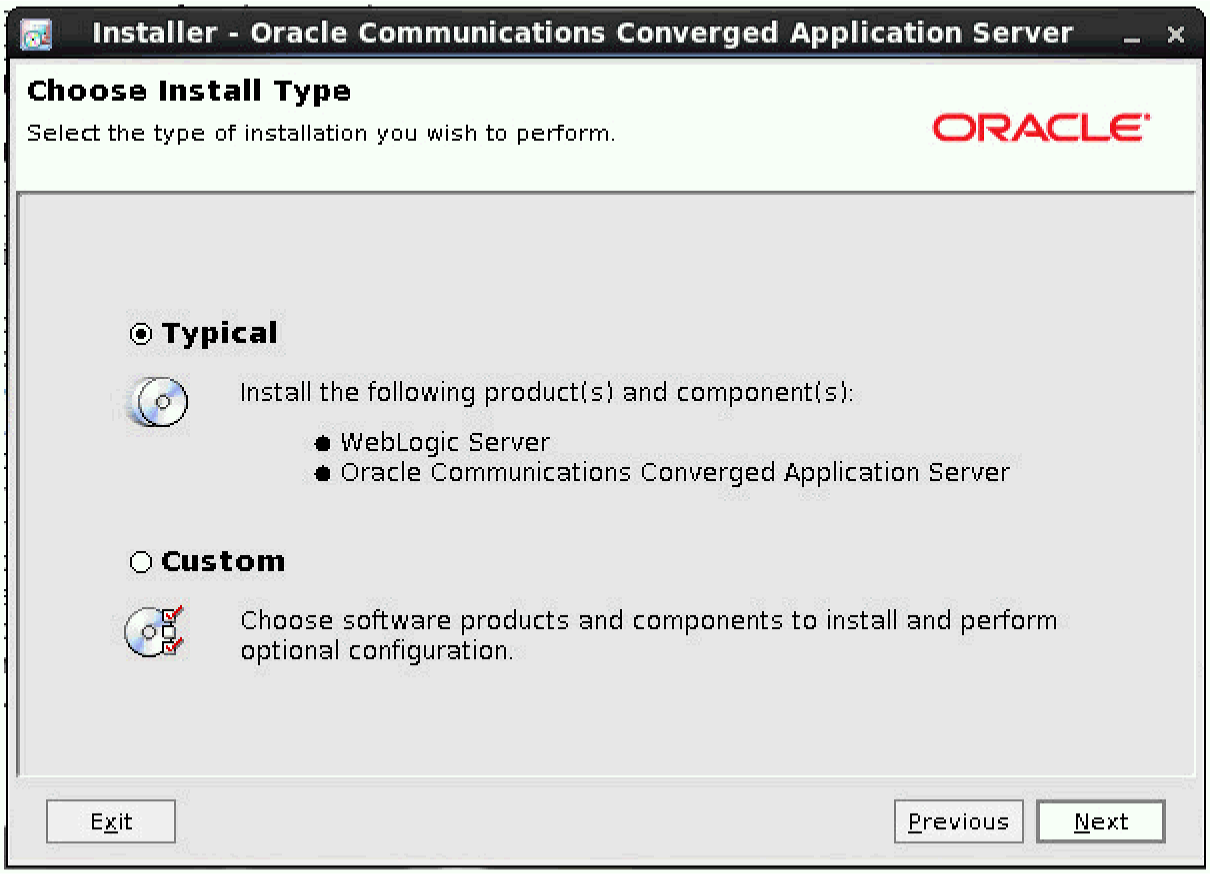
Proxy Username:

Proxy Password:

If this machine has no internet connection, select the checkbox below and click Continue.

☐ I wish to remain uninformed of security issues in my configuration or this machine has no Internet access.

9. Select **Typical,** and then click **Next.**



Installer - Oracle Communications Converged Application Server

Choose Install Type

Select the type of installation you wish to perform.

◉ **Typical**

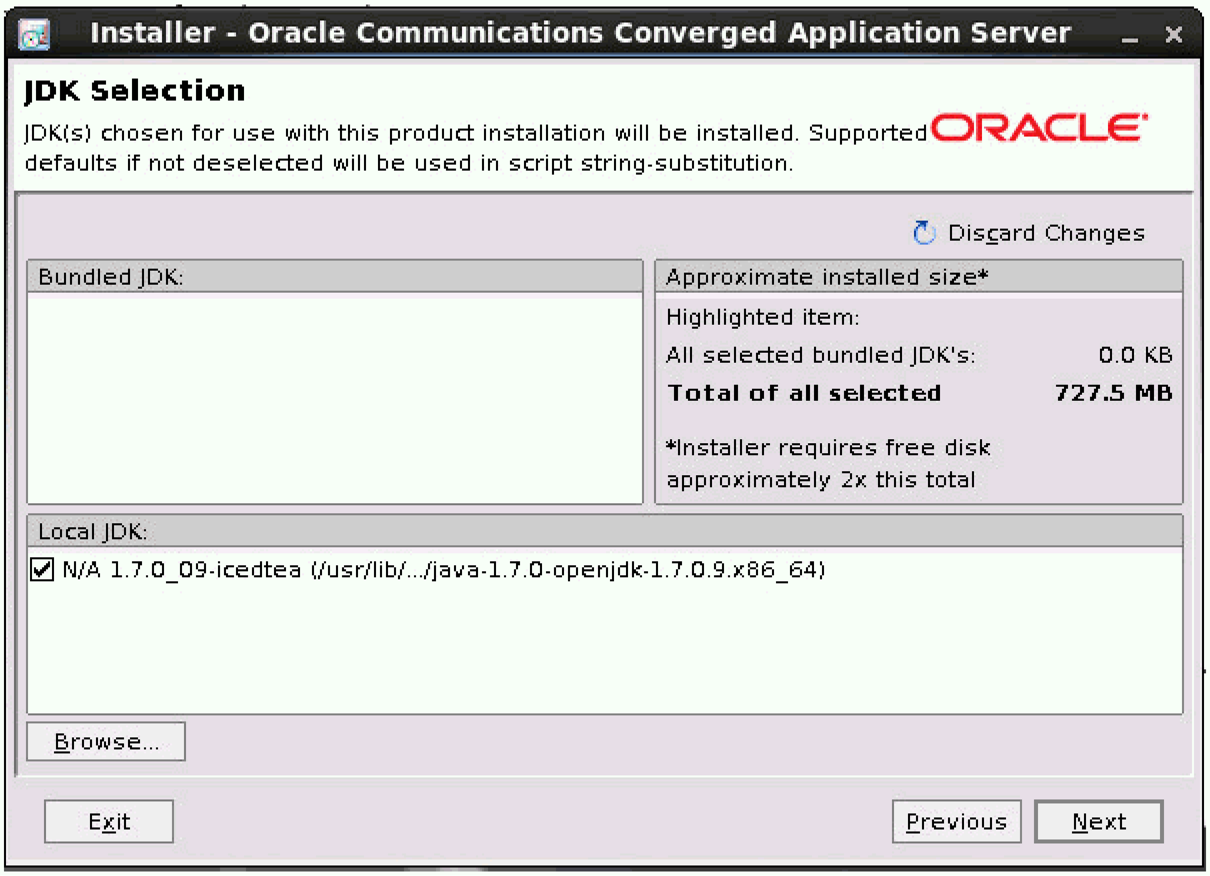
Install the following product(s) and component(s):

- WebLogic Server
- Oracle Communications Converged Application Server

○ **Custom**

Choose software products and components to install and perform optional configuration.

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


Installer - Oracle Communications Converged Application Server

JDK Selection

JDK(s) chosen for use with this product installation will be installed. Supported defaults if not deselected will be used in script string-substitution.

Bundled JDK:

Approximate installed size*

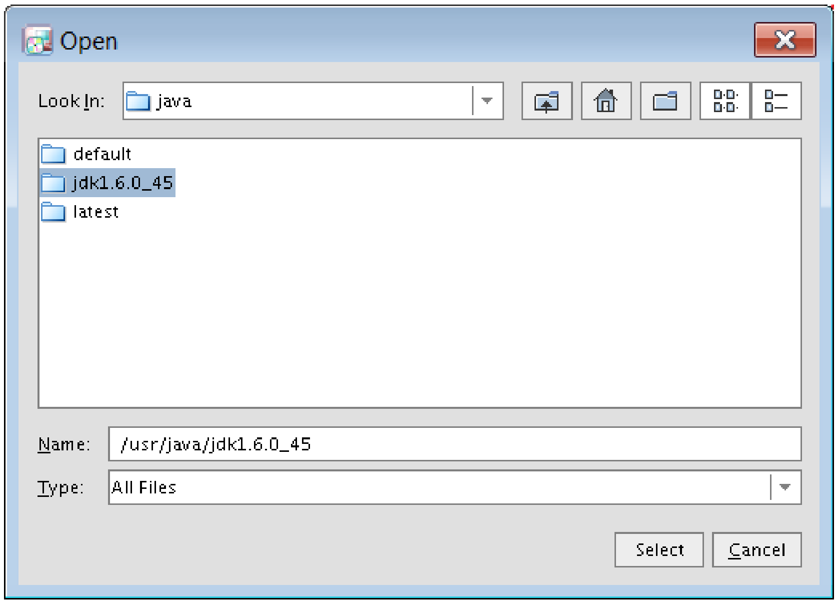
Highlighted item:	
All selected bundled JDK's:	0.0 KB
Total of all selected	**727.5 MB**

*Installer requires free disk approximately 2x this total

Local JDK:

☑ N/A 1.7.0_09-icedtea (/usr/lib/.../java-1.7.0-openjdk-1.7.0.9.x86_64)

11. Select **jdk1.6.0_45** directory, and then click **Select**.



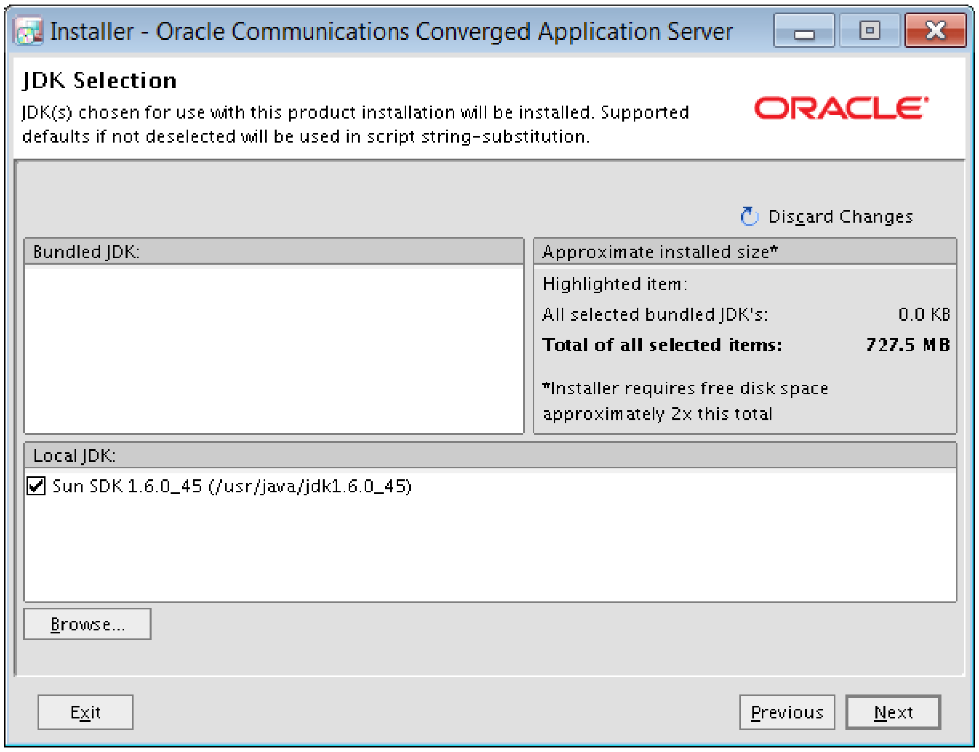
Look In: java

default
jdk1.6.0_45
latest

Name: /usr/java/jdk1.6.0_45

Type: All Files

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



Installer - Oracle Communications Converged Application Server

JDK Selection

JDK(s) chosen for use with this product installation will be installed. Supported defaults if not deselected will be used in script string-substitution.

ORACLE

Discard Changes

Bundled JDK:

Approximate installed size*

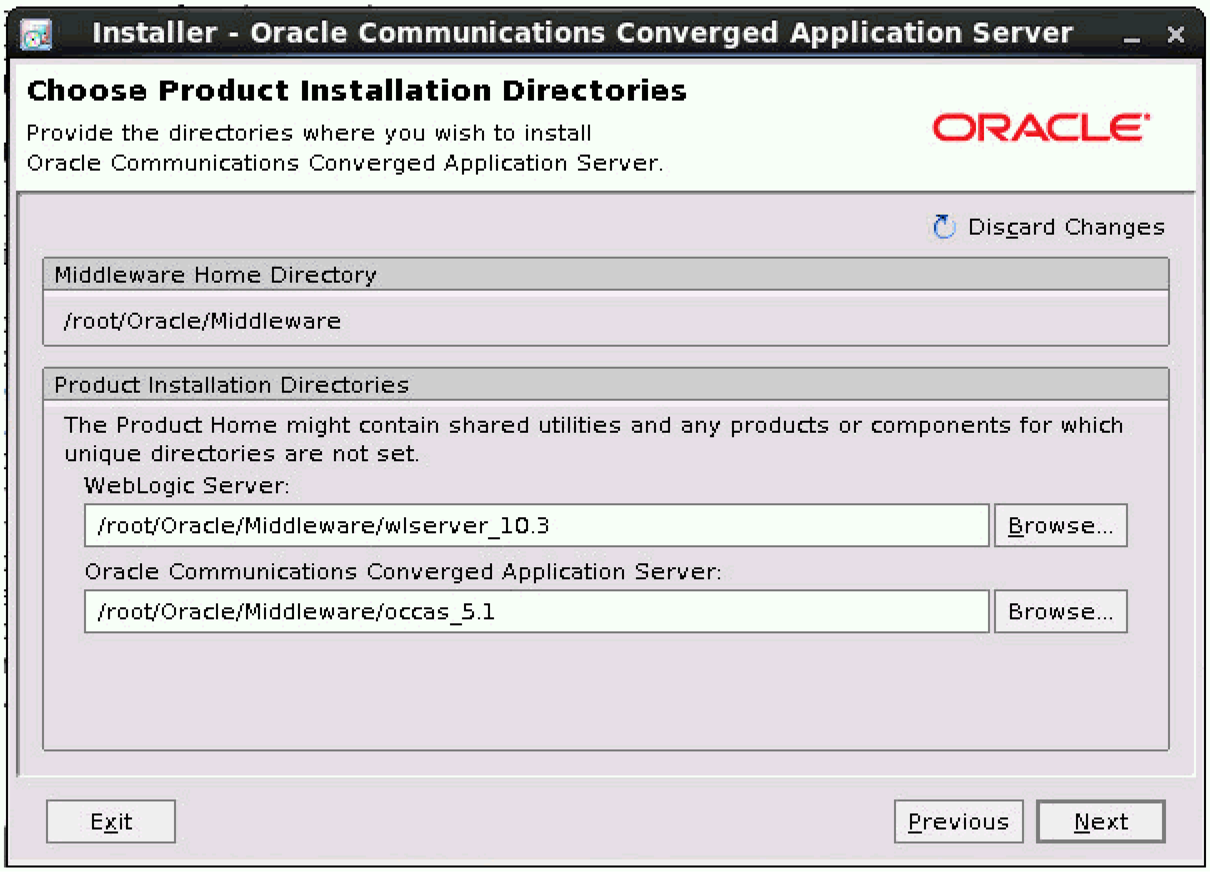
Highlighted item:	
All selected bundled JDK's:	0.0 KB
Total of all selected items:	**727.5 MB**

*Installer requires free disk space approximately 2x this total

Local JDK:

☑ Sun SDK 1.6.0_45 (/usr/java/jdk1.6.0_45)

13. Click **Next**.



Installer - Oracle Communications Converged Application Server

Choose Product Installation Directories

Provide the directories where you wish to install
Oracle Communications Converged Application Server.

Middleware Home Directory

/root/Oracle/Middleware

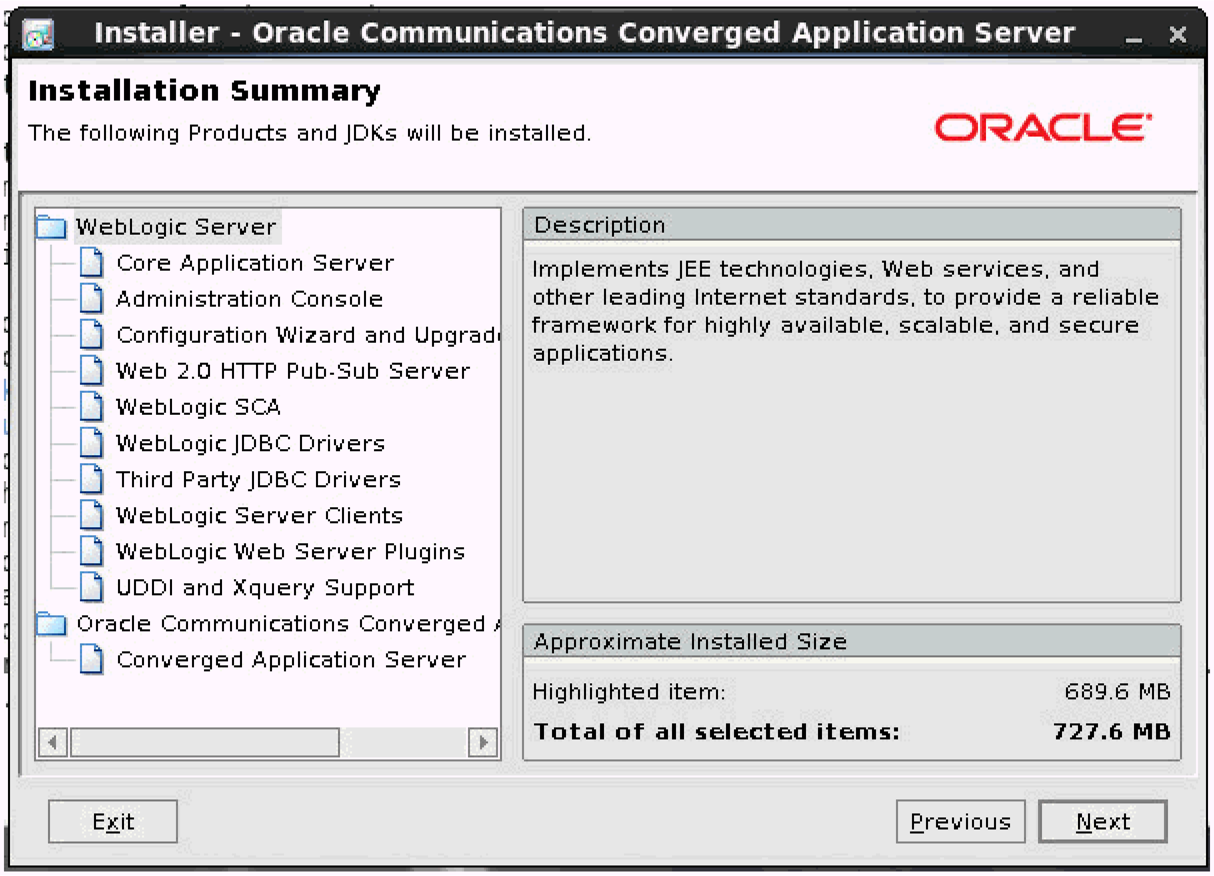
Product Installation Directories

The Product Home might contain shared utilities and any products or components for which unique directories are not set.

WebLogic Server: /root/Oracle/Middleware/wlserver_10.3

Oracle Communications Converged Application Server: /root/Oracle/Middleware/occas_5.1

14. Click **Next**.



Installer - Oracle Communications Converged Application Server

Installation Summary

The following Products and JDKs will be installed.

- WebLogic Server
  - Core Application Server
  - Administration Console
  - Configuration Wizard and Upgrad
  - Web 2.0 HTTP Pub-Sub Server
  - WebLogic SCA
  - WebLogic JDBC Drivers
  - Third Party JDBC Drivers
  - WebLogic Server Clients
  - WebLogic Web Server Plugins
  - UDDI and Xquery Support
- Oracle Communications Converged
  - Converged Application Server

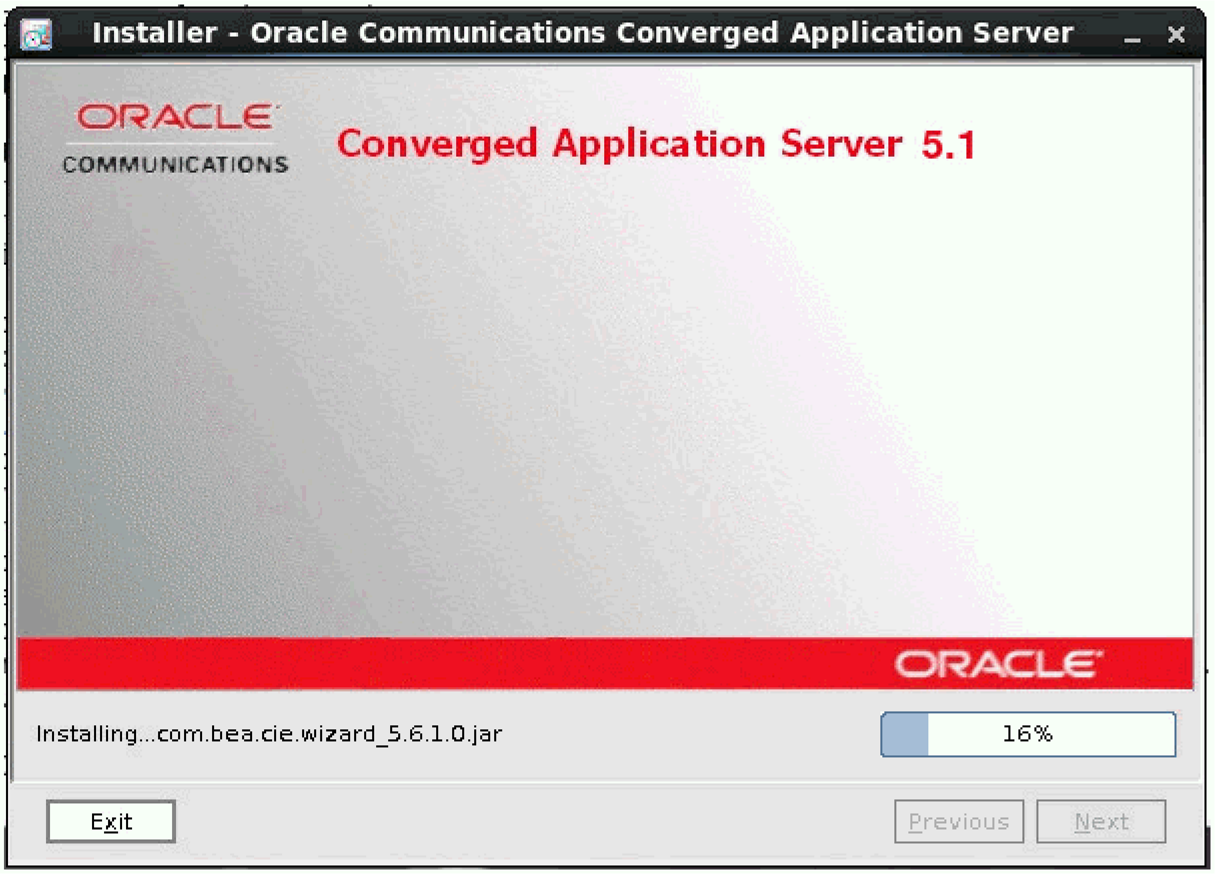
Description

Implements JEE technologies, Web services, and other leading Internet standards, to provide a reliable framework for highly available, scalable, and secure applications.

Approximate Installed Size

Highlighted item:	689.6 MB
Total of all selected items:	**727.6 MB**

15. Click **Next** once the installer is finished to proceed to OCCAS Configuration.


Installer - Oracle Communications Converged Application Server
ORACLE
COMMUNICATIONS
Converged Application Server 5.1
ORACLE
Exit
Previous
Next

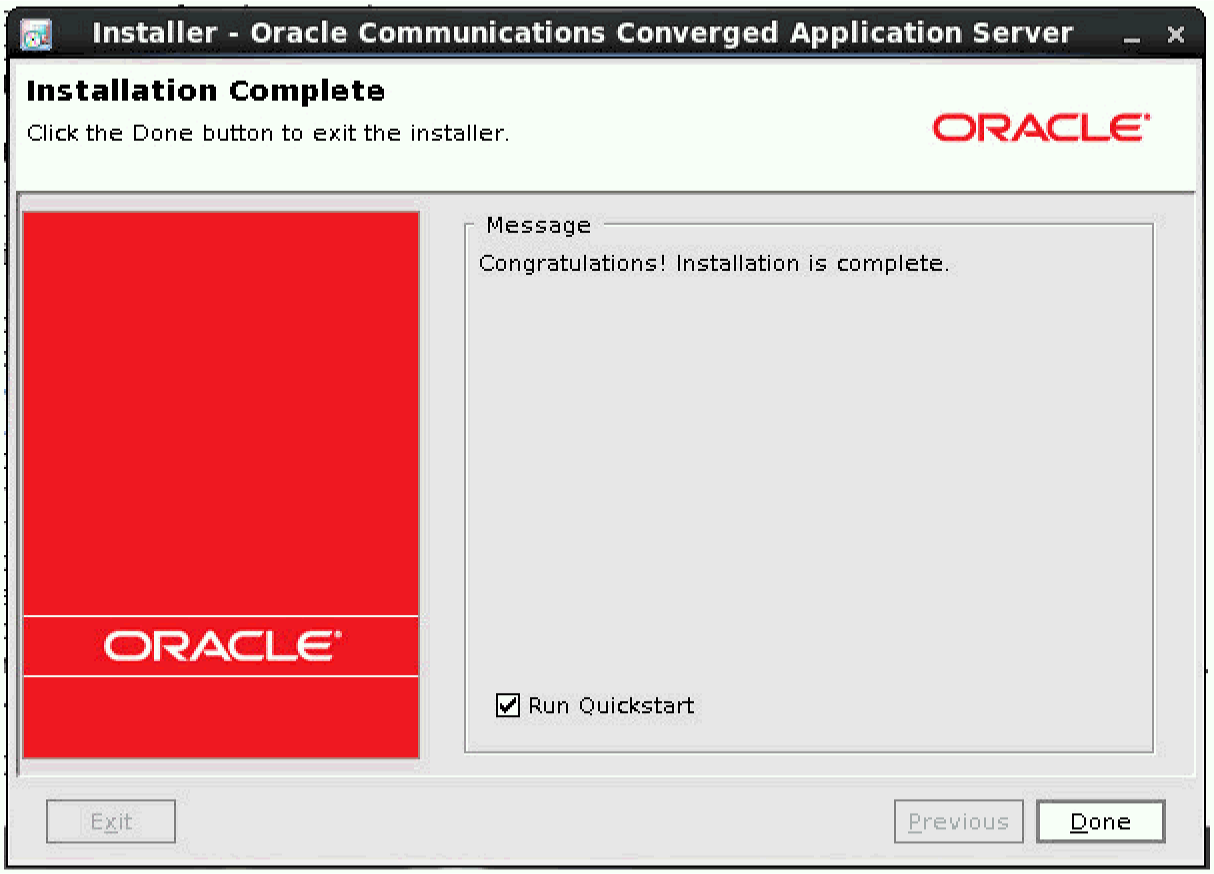
Installing...com.bea.cie.wizard_5.6.1.0.jar

16%

OCCAS Configuration

Proceed as follows to configure OCCAS after installation:

1. Select **Run Quickstart,** and then click **Done.**



Installer - Oracle Communications Converged Application Server

Installation Complete

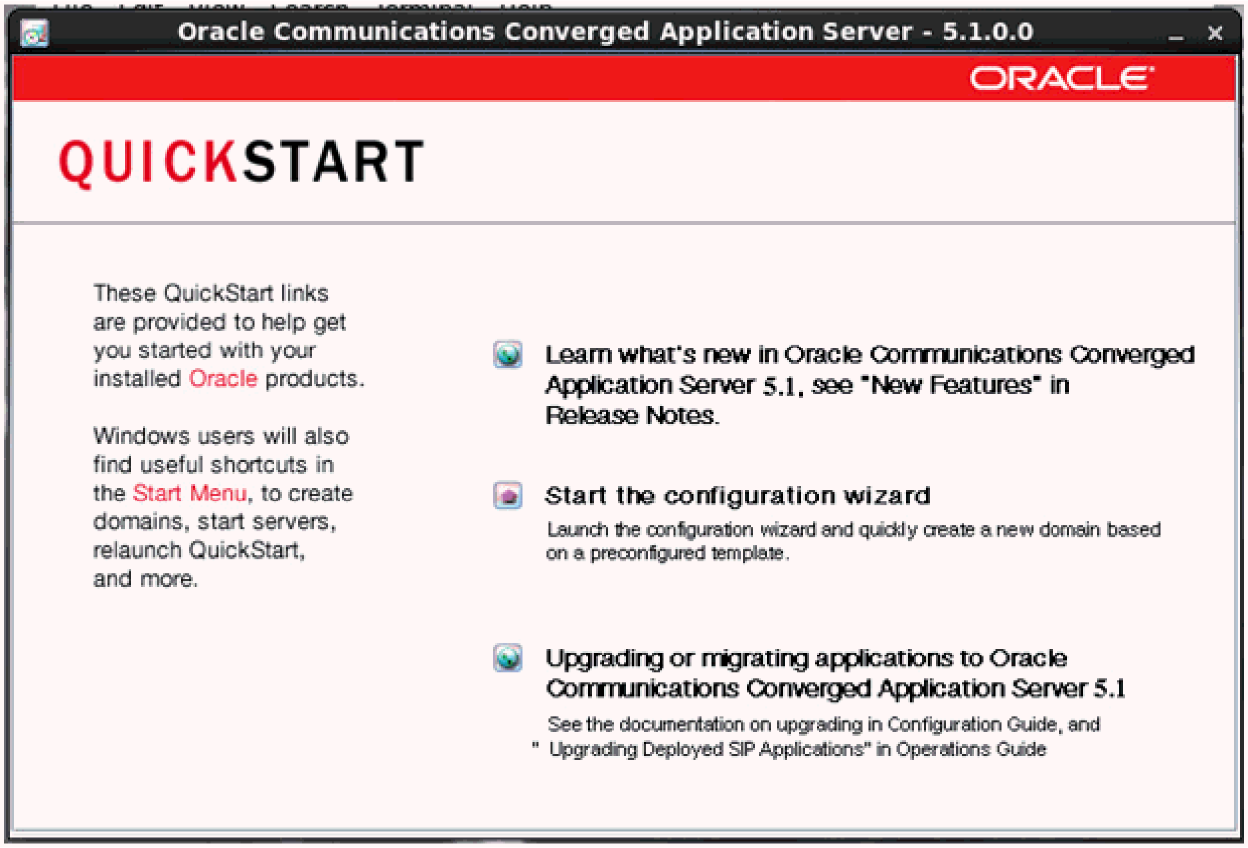
Click the Done button to exit the installer.

Message

Congratulations! Installation is complete.

☑ Run Quickstart

2. Click **Start the configuration wizard**.



Oracle Communications Converged Application Server - 5.1.0.0

QUICKSTART

These QuickStart links are provided to help get you started with your installed Oracle products.

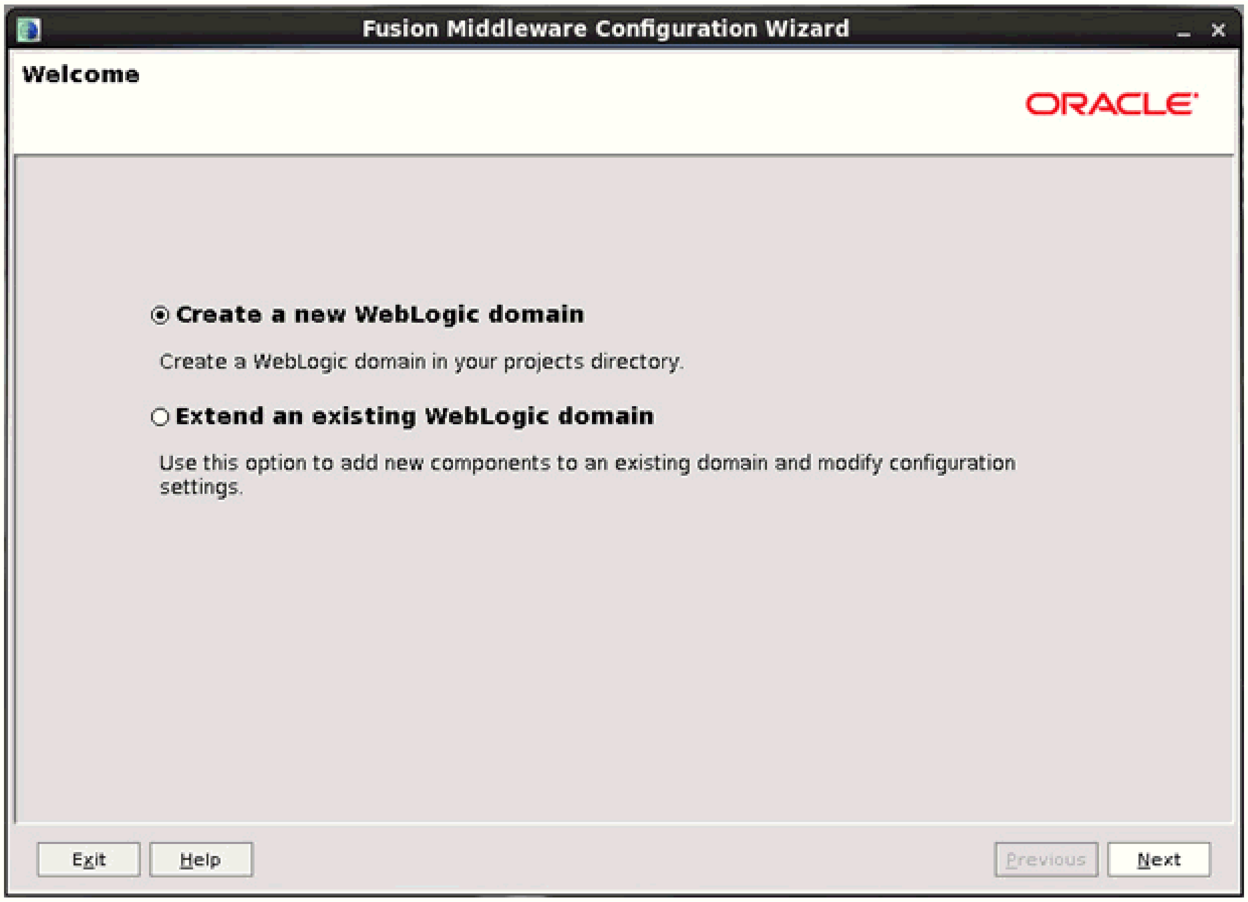
Windows users will also find useful shortcuts in the Start Menu, to create domains, start servers, relaunch QuickStart, and more.

Learn what's new in Oracle Communications Converged Application Server 5.1, see "New Features" in Release Notes.

Start the configuration wizard
Launch the configuration wizard and quickly create a new domain based on a preconfigured template.

Upgrading or migrating applications to Oracle Communications Converged Application Server 5.1
See the documentation on upgrading in Configuration Guide, and " Upgrading Deployed SIP Applications" in Operations Guide

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



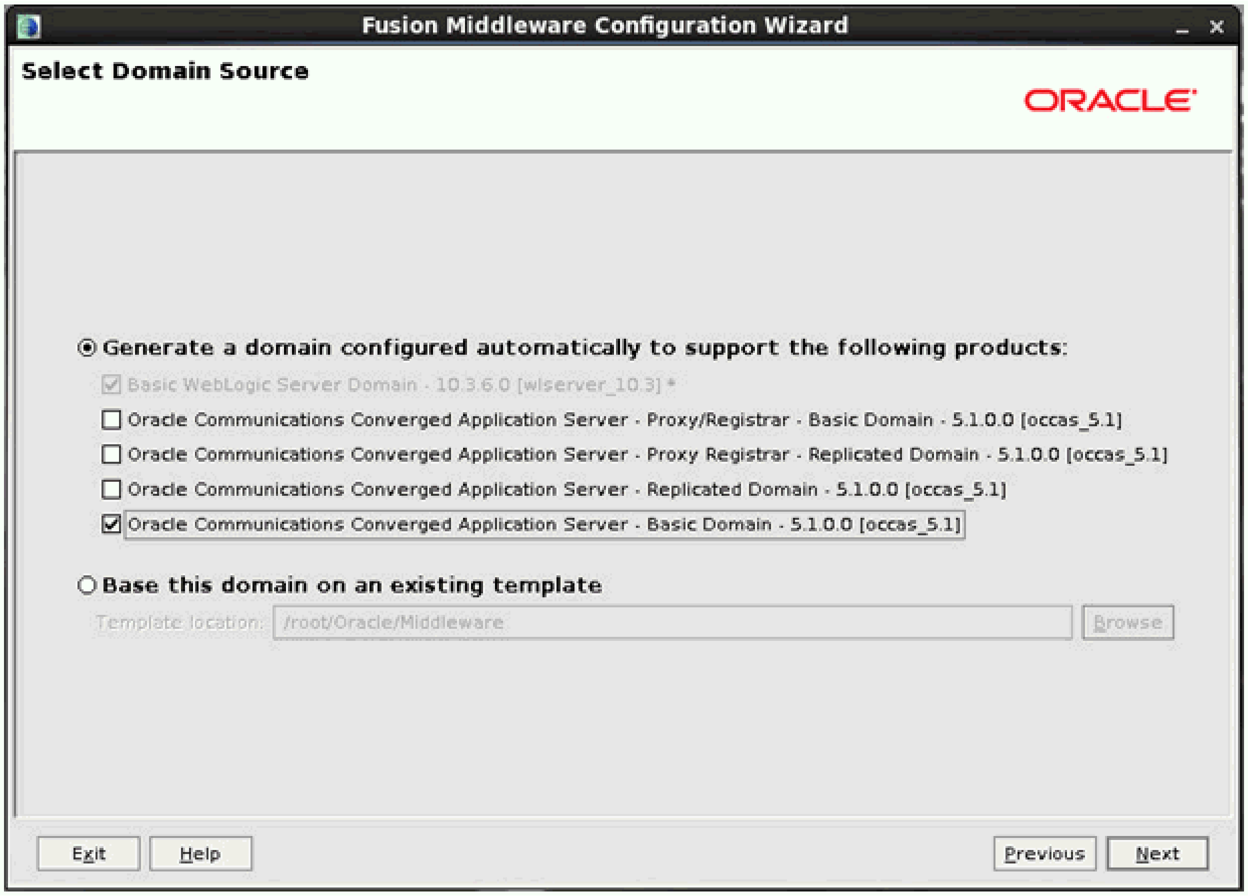
[✓ Create a new WebLogic domain]

Create a WebLogic domain in your projects directory.

Extend an existing WebLogic domain

Use this option to add new components to an existing domain and modify configuration settings.

4. Select **Generate a domain configured automatically to support the following products** and select **Oracle Communications Converged Application Server - Basic Domain - 5.1.0.0 (occas_5.1)**. Then, click **Next**.



Select Domain Source

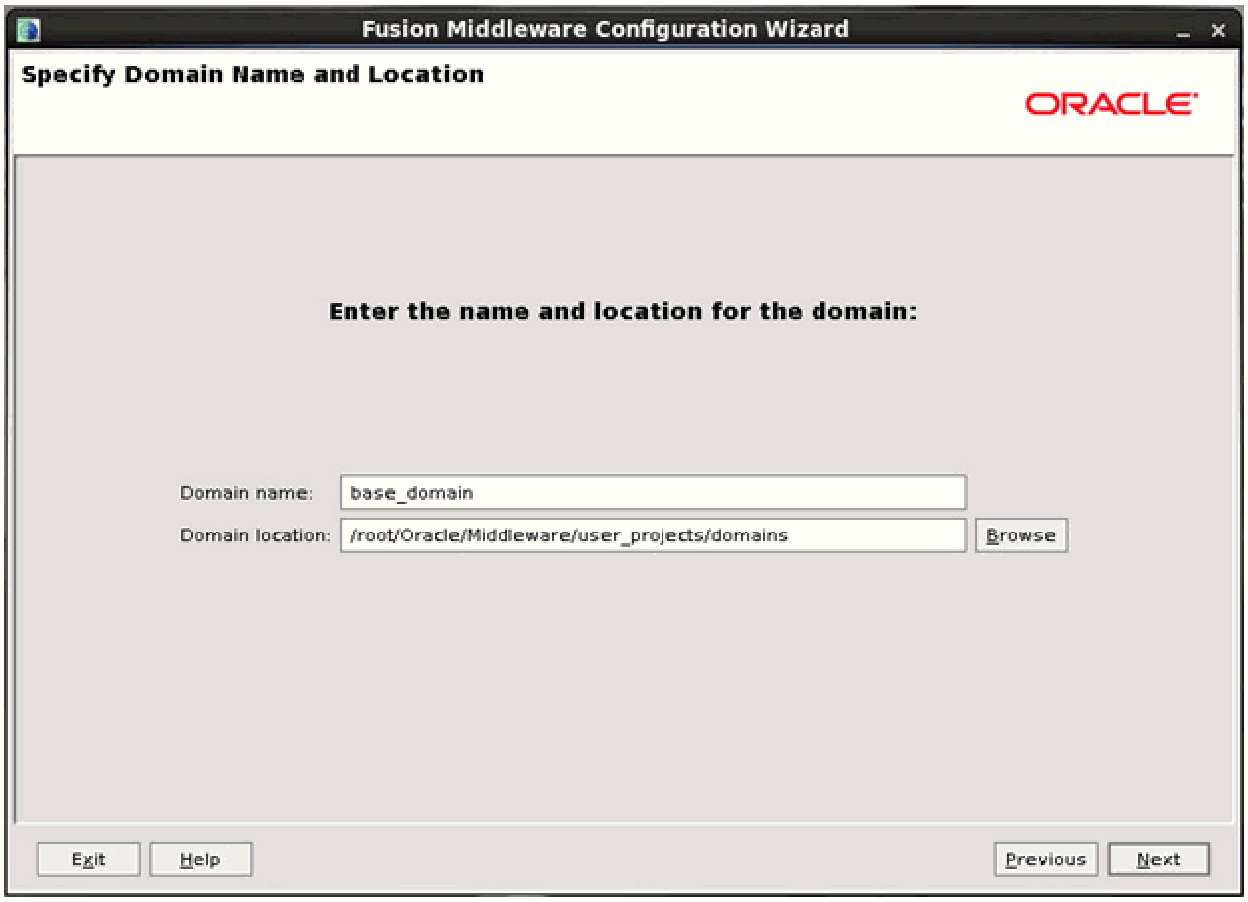
(●) Generate a domain configured automatically to support the following products:

- [✓] Basic WebLogic Server Domain - 10.3.6.0 [wlserver_10.3] *
- [ ] Oracle Communications Converged Application Server - Proxy/Registrar - Basic Domain - 5.1.0.0 [occas_5.1]
- [ ] Oracle Communications Converged Application Server - Proxy Registrar - Replicated Domain - 5.1.0.0 [occas_5.1]
- [ ] Oracle Communications Converged Application Server - Replicated Domain - 5.1.0.0 [occas_5.1]
- [✓] Oracle Communications Converged Application Server - Basic Domain - 5.1.0.0 [occas_5.1]

( ) Base this domain on an existing template

Template location: /root/Oracle/Middleware

5. Click **Next**.



Fusion Middleware Configuration Wizard

Specify Domain Name and Location

Enter the name and location for the domain:

Domain name: base_domain

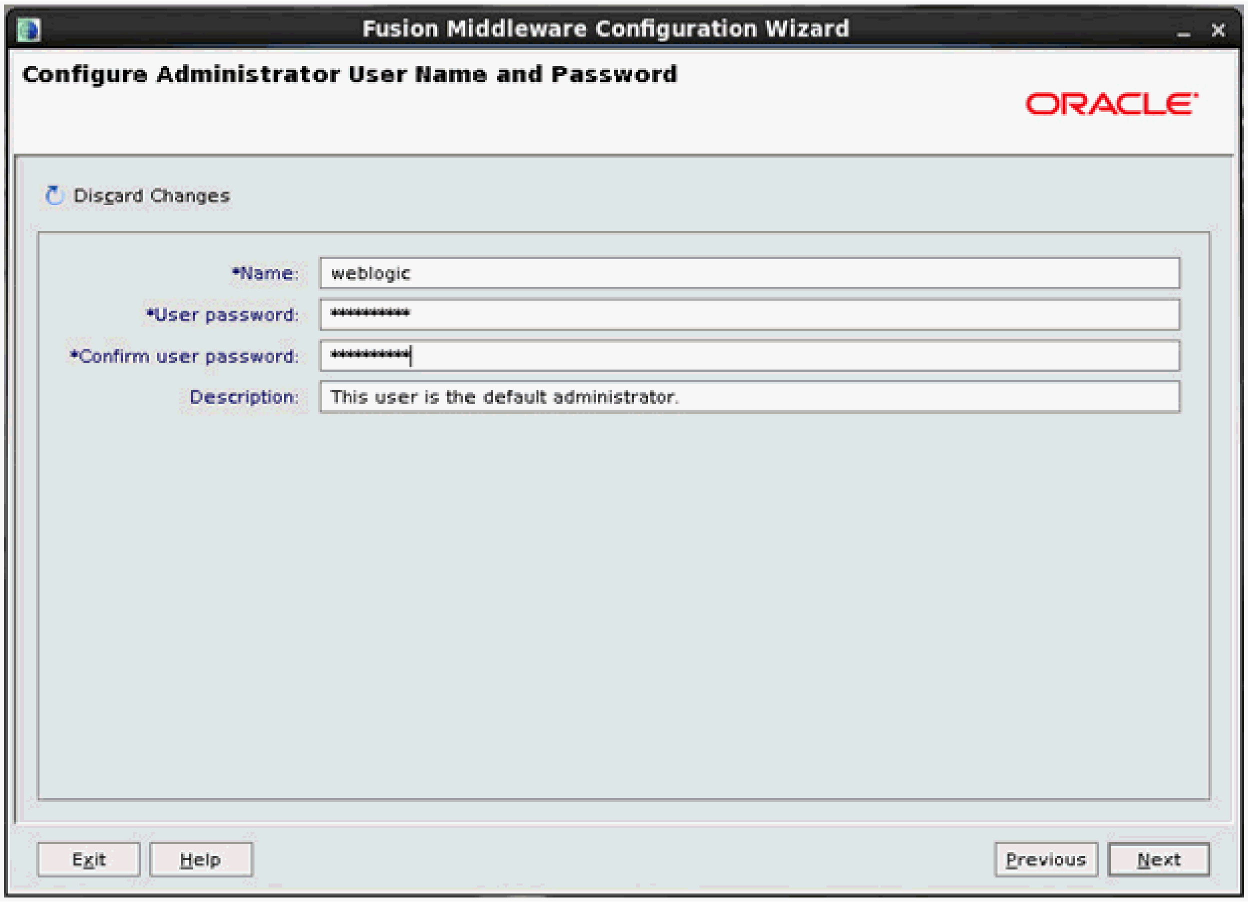
Domain location: /root/Oracle/Middleware/user_projects/domains

6. Specify **Name** and **User password**, and then click **Next**. The following is used as an example:

   Name: weblogic

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

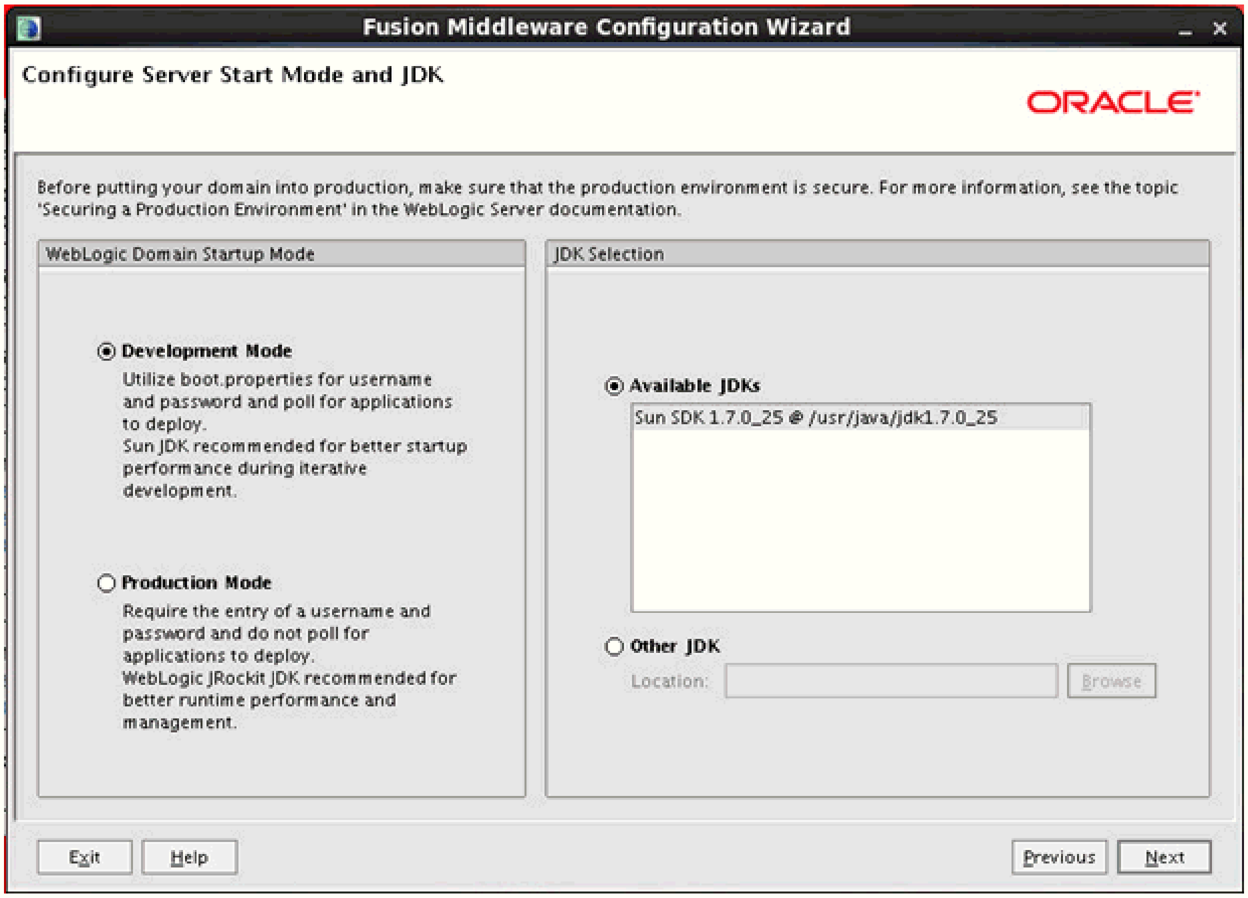
   Note: A strong password is required.



Configure Administrator User Name and Password

*Name: weblogic
*User password: **********
*Confirm user password: **********
Description: This user is the default administrator.

7. Select **Development Mode**, and then click **Next**.



Configure Server Start Mode and JDK

Before putting your domain into production, make sure that the production environment is secure. For more information, see the topic 'Securing a Production Environment' in the WebLogic Server documentation.

WebLogic Domain Startup Mode

[✓ Development Mode]
Utilize boot.properties for username and password and poll for applications to deploy.
Sun JDK recommended for better startup performance during iterative development.

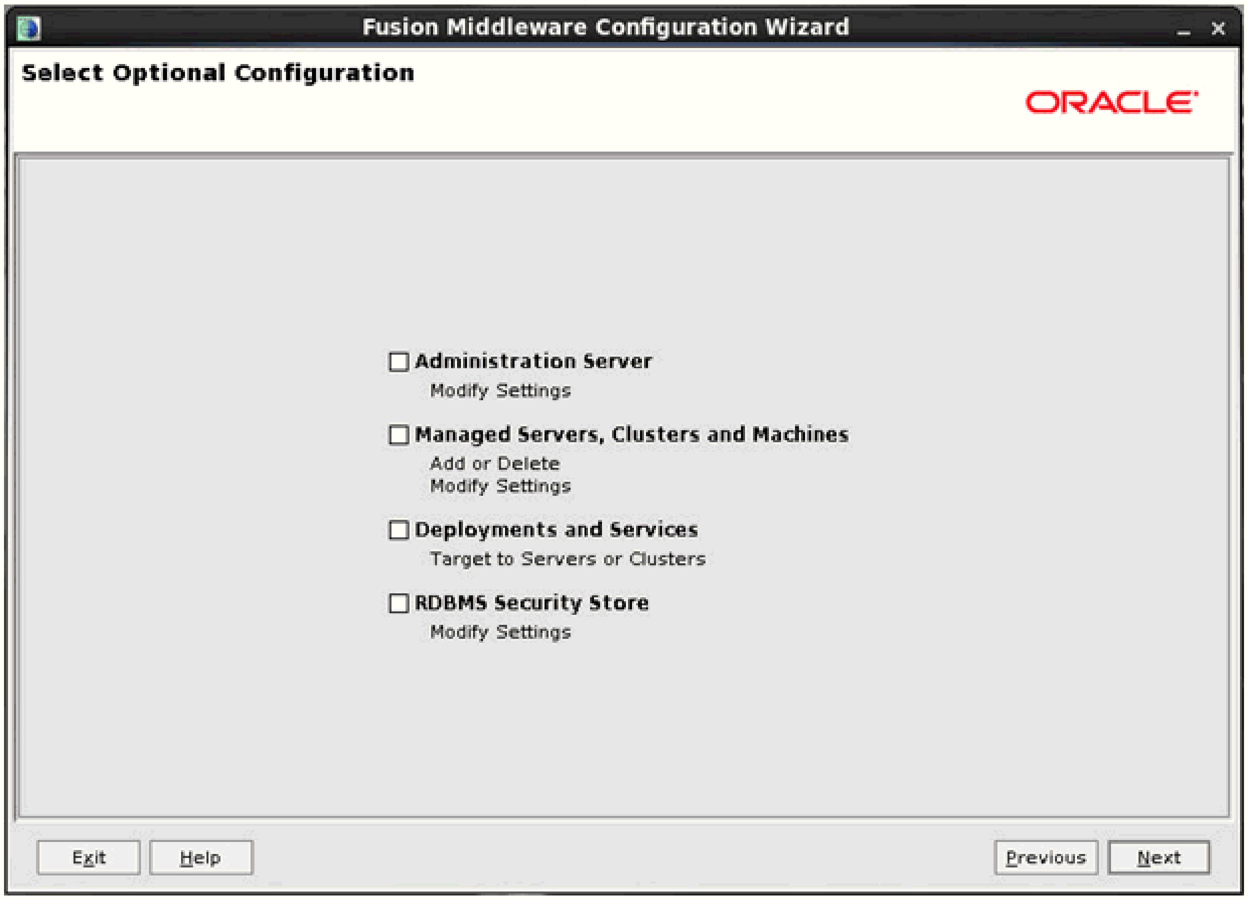
Production Mode
Require the entry of a username and password and do not poll for applications to deploy.
WebLogic JRockit JDK recommended for better runtime performance and management.

JDK Selection

[✓ Available JDKs]
Sun SDK 1.7.0_25 @ /usr/java/jdk1.7.0_25

Other JDK
Location:

8. Click **Next**.



Select Optional Configuration

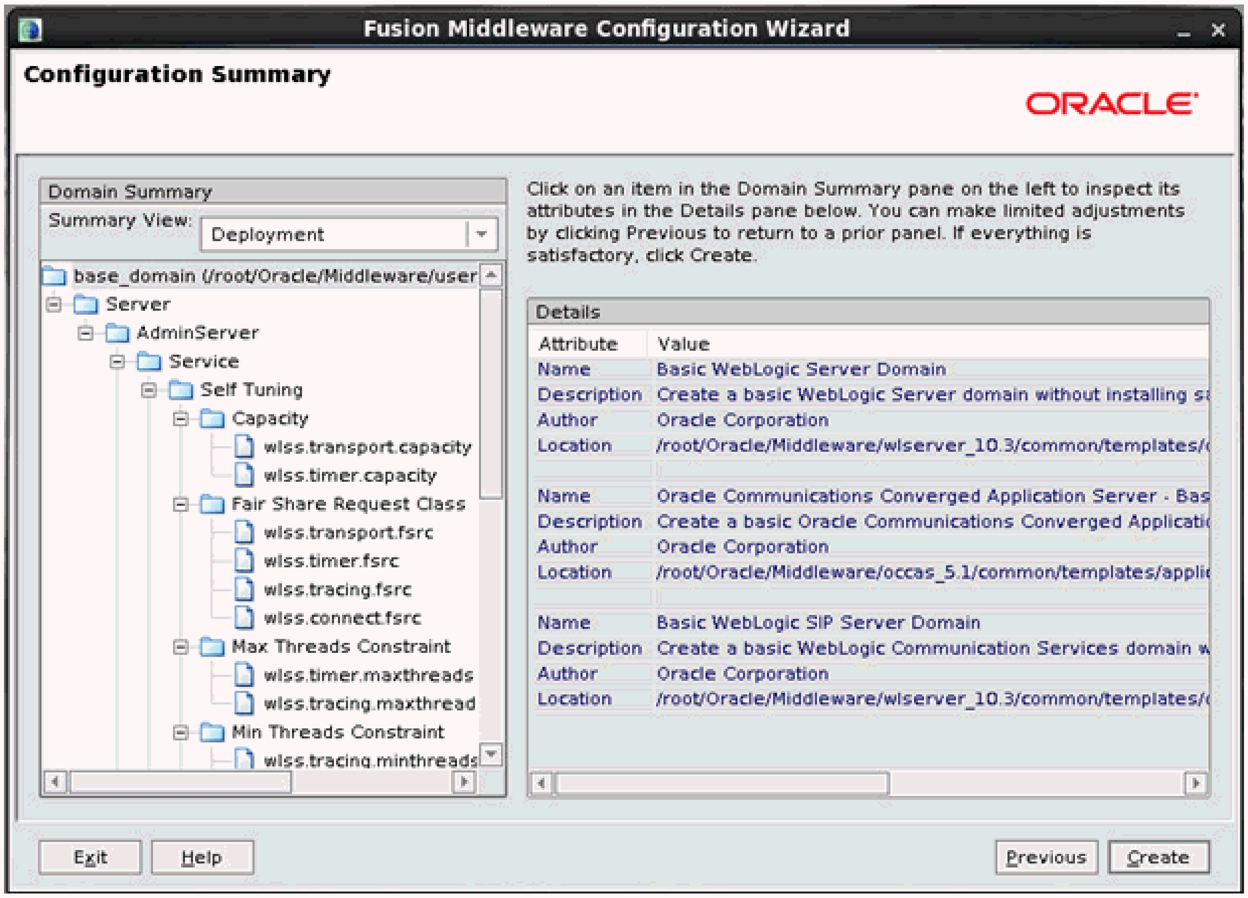
☐ Administration Server
Modify Settings

☐ Managed Servers, Clusters and Machines
Add or Delete
Modify Settings

☐ Deployments and Services
Target to Servers or Clusters

☐ RDBMS Security Store
Modify Settings

9. Click **Create**.



Fusion Middleware Configuration Wizard

Configuration Summary

Domain Summary

Summary View: Deployment

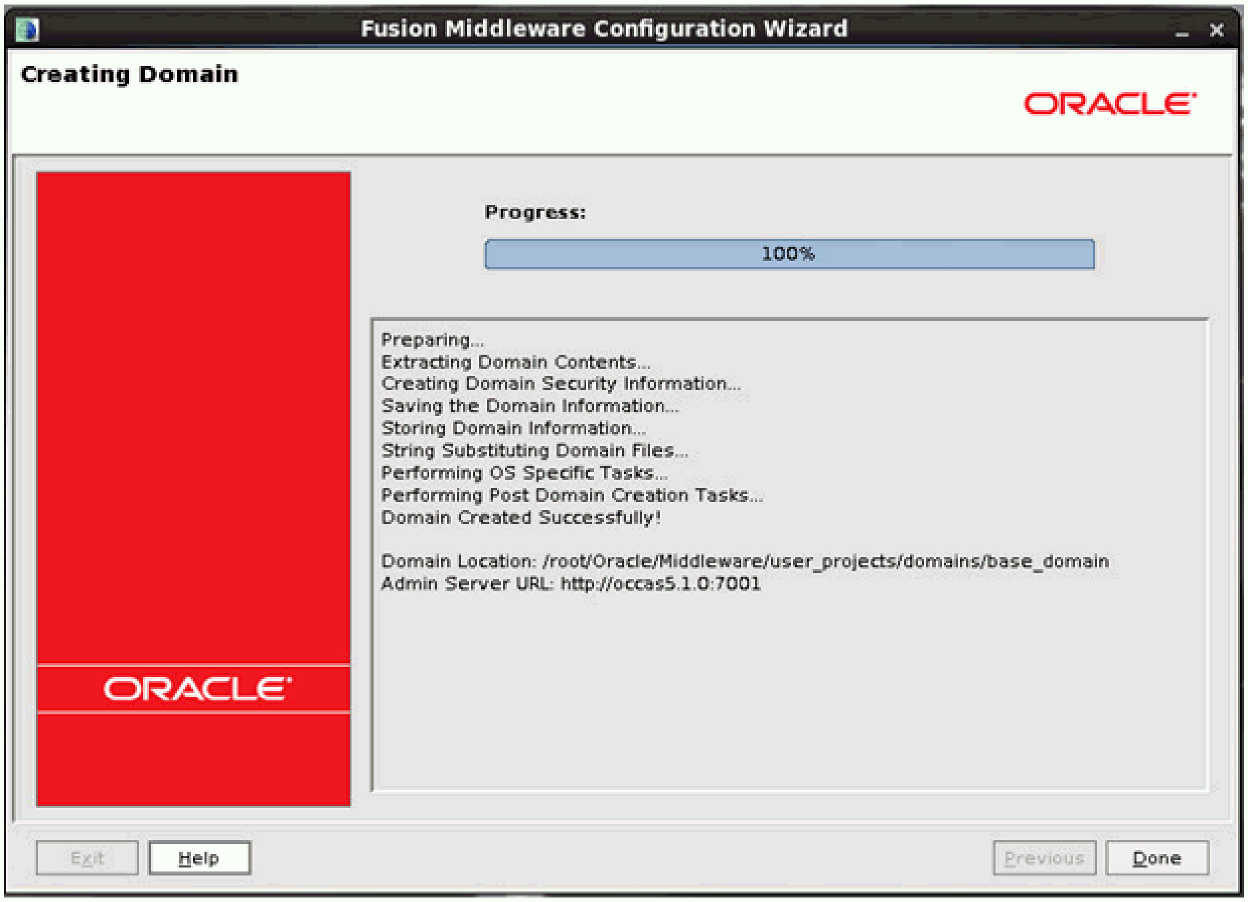
```
base_domain (/root/Oracle/Middleware/user
⊟ Server
  ⊟ AdminServer
    ⊟ Service
      ⊟ Self Tuning
        ⊟ Capacity
            wlss.transport.capacity
            wlss.timer.capacity
        ⊟ Fair Share Request Class
            wlss.transport.fsrc
            wlss.timer.fsrc
            wlss.tracing.fsrc
            wlss.connect.fsrc
        ⊟ Max Threads Constraint
            wlss.timer.maxthreads
            wlss.tracing.maxthread
        ⊟ Min Threads Constraint
            wlss.tracing.minthreads
```

Click on an item in the Domain Summary pane on the left to inspect its attributes in the Details pane below. You can make limited adjustments by clicking Previous to return to a prior panel. If everything is satisfactory, click Create.

Details

Attribute	Value
Name	Basic WebLogic Server Domain
Description	Create a basic WebLogic Server domain without installing s
Author	Oracle Corporation
Location	/root/Oracle/Middleware/wlserver_10.3/common/templates/
Name	Oracle Communications Converged Application Server - Bas
Description	Create a basic Oracle Communications Converged Applicati
Author	Oracle Corporation
Location	/root/Oracle/Middleware/occas_5.1/common/templates/appli
Name	Basic WebLogic SIP Server Domain
Description	Create a basic WebLogic Communication Services domain w
Author	Oracle Corporation
Location	/root/Oracle/Middleware/wlserver_10.3/common/templates/

10. When the configuration is finished, click **Done**. The OCCAS installation and configuration are now complete.



```
Preparing...
Extracting Domain Contents...
Creating Domain Security Information...
Saving the Domain Information...
Storing Domain Information...
String Substituting Domain Files...
Performing OS Specific Tasks...
Performing Post Domain Creation Tasks...
Domain Created Successfully!

Domain Location: /root/Oracle/Middleware/user_projects/domains/base_domain
Admin Server URL: http://occas5.1.0:7001
```

OCCAS Startup

To start OCCAS 5.1, go to the *<Domain Location>/bin* directory:

```
/home/occas5.1/Oracle/Middleware/user_projects/domains/base_domain/bin
```

Run the following command:

```
./startWebLogic.sh
```

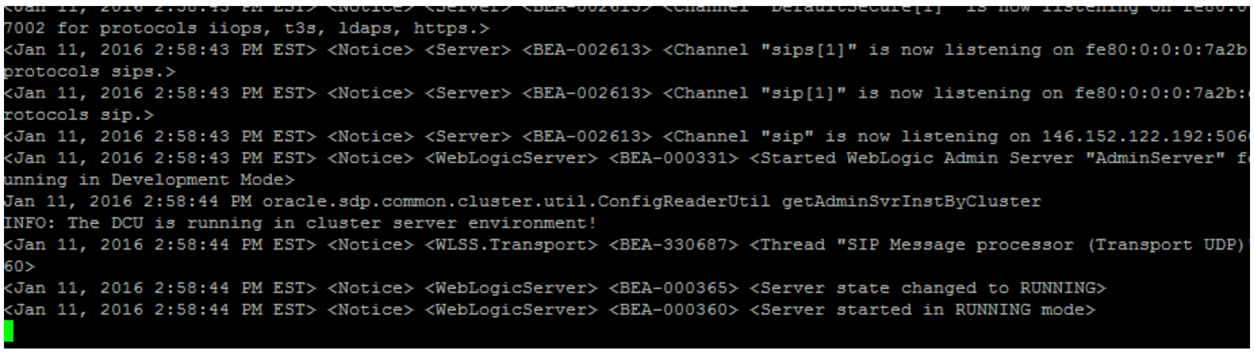
Since the Development Mode installation was chosen, it is not necessary to enter the username/password during script startup. If the Production Mode installation was chosen, it is necessary to specify username/password.

The following is an example:

Name: weblogic

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

To verify that OCCAS 5.1 is started, check if **<Server started in RUNNING mode>** is displayed.



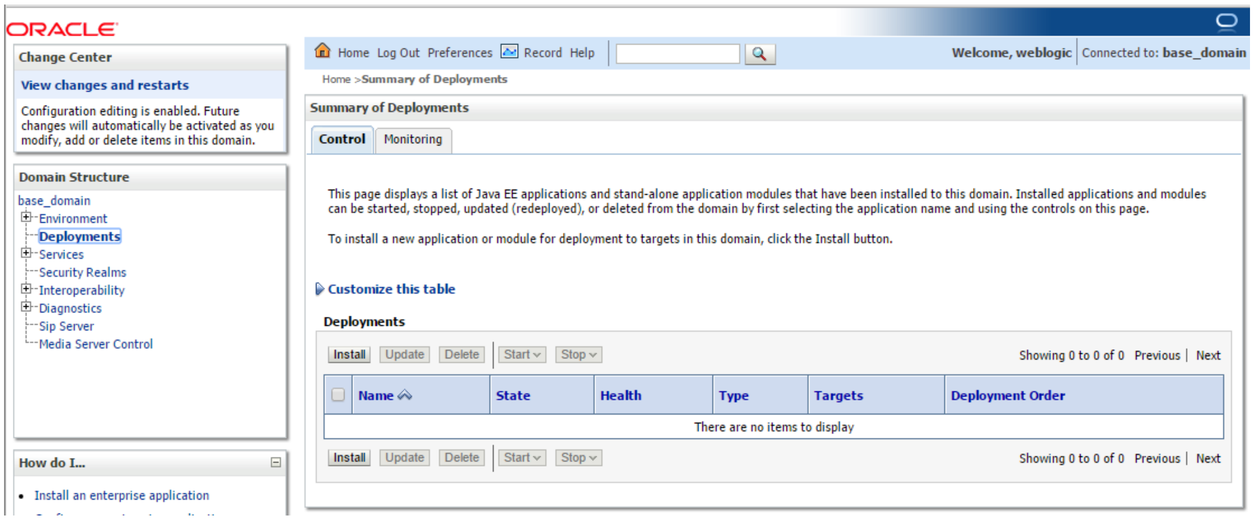
```
7002 for protocols iiops, t3s, ldaps, https.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sips[1]" is now listening on fe80:0:0:0:7a2b
protocols sips.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sip[1]" is now listening on fe80:0:0:0:7a2b:
rotocols sip.>
<Jan 11, 2016 2:58:43 PM EST> <Notice> <Server> <BEA-002613> <Channel "sip" is now listening on 146.152.122.192:506
<Jan 11, 2016 2:58:43 PM EST> <Notice> <WebLogicServer> <BEA-000331> <Started WebLogic Admin Server "AdminServer" f
unning in Development Mode>
Jan 11, 2016 2:58:44 PM oracle.sdp.common.cluster.util.ConfigReaderUtil getAdminSvrInstByCluster
INFO: The DCU is running in cluster server environment!
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WLSS.Transport> <BEA-330687> <Thread "SIP Message processor (Transport UDP)
60>
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to RUNNING>
<Jan 11, 2016 2:58:44 PM EST> <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>
```

Firewall Configuration

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

OCCAS Verification

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


ORACLE
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of Deployments

Change Center

View changes and restarts

Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

Domain Structure

- base_domain
  - Environment
  - Deployments
  - Services
  - Security Realms
  - Interoperability
  - Diagnostics
  - Sip Server
  - Media Server Control

How do I...

- Install an enterprise application

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

Install Update Delete Start Stop — Showing 0 to 0 of 0 Previous | Next

Name	State	Health	Type	Targets	Deployment Order
There are no items to display					

Install Update Delete Start Stop — Showing 0 to 0 of 0 Previous | Next

8. Appendix B: Updating the Dialogic JSR 309 Connector

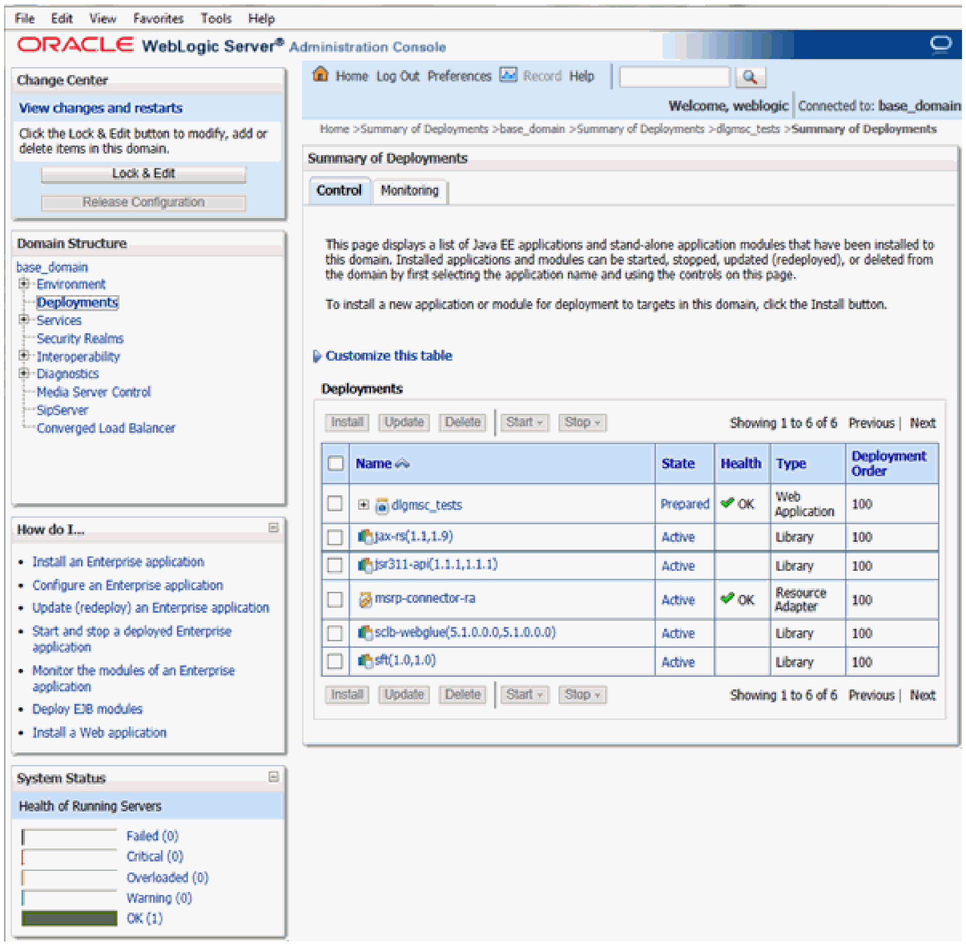
The Dialogic JSR 309 Connector comes as a set of JAVA library (JAR) files. In the OCCAS Application Server, the required application files are stored as part of the Application Server configuration and are located in the *lib* directory of the OCCAS *DOMAIN_HOME* directory.

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

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

- *dialogic309-5.0.xxxx-occas5.1.jar*
- *dialogicmsmltypes-5.0.xxxx-occas5.1.jar*
- *dialogicsmiltypes-5.0-xxxx-occas5.1.jar*

In order for the new JAVA library JAR file to take effect, stop and restart the application using it using the OCCAS administration page (**Deployments** under **Domain Structure**) as shown below.


File Edit View Favorites Tools Help
ORACLE WebLogic Server® Administration Console
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: base_domain
Home >Summary of Deployments >base_domain >Summary of Deployments >dlgmsc_tests >Summary of Deployments
Domain Structure
base_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
Media Server Control
SipServer
Converged Load Balancer
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
Deploy EJB modules
Install a Web application
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (1)
Summary of Deployments
Control
Monitoring
Customize this table
Install Update Delete Start Stop
Showing 1 to 6 of 6 Previous | Next

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

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.

Deployments

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