Oracle® WebLogic Communication Services

Administrator's Guide

11*g* Release 1 (11.1.1)

E13806-01

May 2009

ORACLE®

Oracle WebLogic Communication Services Administrator's Guide, 11*g* Release 1 (11.1.1)

E13806-01

Contents

6 Configuring SIP Data Tier Partitions and Replicas

7 Provisioning Users With Sash

8 Monitoring and Troubleshooting

12 Monitoring Oracle User Messaging Service

List of Examples

List of Figures

List of Tables

Preface

This book details conceptual, topology and configuration topics about Oracle WebLogic Communication Services. This Preface includes the following topics:

Audience

The intended audience is system administrators who will set up and maintain Oracle WebLogic Communication Services for their organization.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle

technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

Related Documents

For more information, see the following documents in the documentation set:

- *Oracle WebLogic Communication Services Installation Guide*
- *Oracle WebLogic Communication Services Developer's Guide*
- *Oracle Fusion Middleware 11g Release Notes*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

General Configuration

This Part contains the following chapters:

1

Introduction to Oracle WebLogic Communication Services

The following section provides an introduction to Oracle WebLogic Communication Services (OWLCS), and an overview of how to configure and manage Oracle WebLogic Communication Services deployments:

- Section 1.1, "Oracle WebLogic Communication Services"

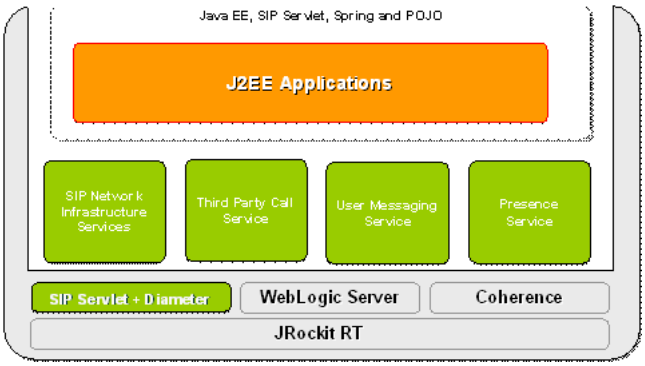
1.1 Oracle WebLogic Communication Services

Oracle WebLogic Communication Services 11*g* (OWLCS) is a comprehensive platform designed to integrate communication services with enterprise services and applications. It includes easy to consume services to support interactions with key communication channels. OWLCS supports the following technologies.

- Section 1.1.1, "Messaging"
- Section 1.1.2, "Telephony"
- Section 1.1.3, "Presence"
- Section 1.1.4, "WebLogic Server 10.3 Platform with support for SIP and converged applications"

Figure 1–1 shows a simplified overview of Oracle WebLogic Communication Services; more details will be presented in later chapters.

Figure 1–1 OWLCS overview



1.1.1 Messaging

OWLCS supports a simple and reliable way of integrating multi channel messaging in to applications through its User Messaging Service (UMS). UMS supports both Java API's as well as Web-Services for integration. The channels supported include SMS, E-Mail, Instant Messaging as well as Voice messages. UMS also supports intelligent messaging whereby the final destination of a message is determined by a users preferences.

For more information on installing and configuring UMS, see Chapter 11, "Configuring Oracle User Messaging Service".

For more information on developing messaging applications, see *Oracle WebLogic Communication Services Developer's Guide*.

1.1.2 Telephony

OWLCS supports a Third Party Call Control (TPCC) Web-Services API for applications to initiate Voice over IP calls between two phones or soft clients. This service integrates out of the box with market leading IP-based PBX-systems as well as the PSTN using VoIP-PSTN gateways.

For more information on installing and configuring TPCC and creating TPCC applications, see "Third Party Call Service" in *Oracle WebLogic Communication Services Developer's Guide*.

1.1.3 Presence

OWLCS includes a Presence Service that acts as the aggregator of presence information and provides a subscribe/notify paradigm for applications and end-users to consume presence information. An application can integrate either using Web-Services or by using a compliant SIP-based end-user client.

For more information on configuring Presence, see Chapter 9, "Configuring Presence and Presence Web Services". For more information on developing Presence applications, see "ParlayX Presence Web Services" in *Oracle WebLogic Communication Services Developer's Guide*.

1.1.4 WebLogic Server 10.3 Platform with support for SIP and converged applications

OWLCS extends the core WebLogic Server platform with a SIP Container compliant with JSR 289. This enables the development of J2EE applications that processes SIP in addition to HTTP for any advanced communications application. The platform enables the development of complementary communications services that integrate with SIP-based IP-PBXs as well as other SIP elements such as standard SIP clients.

Out of the box, OWLCS provides the key infrastructure applications to build a SIP-based network.

For more information on installing and configuring the SIP Container and about SIP infrastructure applications, see Part I, "General Configuration" and Part V, "Confiding SIP Infrastructure Applications".

2

Shared Configuration Tasks

The following sections provide an overview of the configuration tasks that are common to both Oracle WebLogic Communication Services and Oracle WebLogic Server. These topics are included:

- Section 2.1, "Shared Configuration Tasks for Oracle WebLogic Communication Services and Oracle WebLogic Server"
- Section 2.2, "Oracle WebLogic Communication Services Configuration Overview"
- Section 2.3, "Methods and Tools for Performing Configuration Tasks"
- Section 2.4, "Starting and Stopping Servers"
- Section 2.5, "Administration Server Best Practices"
- Section 2.6, "Common Configuration Tasks"

2.1 Shared Configuration Tasks for Oracle WebLogic Communication Services and Oracle WebLogic Server

Oracle WebLogic Communication Services is based on the Oracle WebLogic Server 10*g* Release 3 application server, and many system-level configuration tasks are the same for both products. This guide addresses only those system-level configuration tasks that are unique to Oracle WebLogic Communication Services, such as tasks related to network and security configuration and cluster configuration for the engine and SIP data tiers.

HTTP server configuration and other basic configuration tasks such as server logging are addressed in Oracle WebLogic Server documentation. See *Oracle Fusion Middleware Getting Started With Installation for Oracle WebLogic Server* to get started.

2.2 Oracle WebLogic Communication Services Configuration Overview

The SIP Servlet container, SIP data tier replication, and Diameter protocol features of Oracle WebLogic Communication Services are implemented in the Oracle WebLogic Server 10*g* Release 3 product as custom resources. A pair of custom resources, `sipserver` and `datatier`, implement the engine tier SIP Servlet container functionality and SIP data tier replication functionality. In production deployments, both resources are generally installed. Specialized deployments may use only the `sipserver` resource in conjunction with a SIP-aware load balancer, as described in Section 15.7, "Alternate Configurations".

Another custom resource, `diameter`, provides Diameter base protocol functionality, and is required only for deployments that utilize one or more Diameter protocol applications.

The Oracle WebLogic Communication Services custom resource assignments are visible in the domain configuration file, `config.xml`, and should not be modified. Example 2–1 shows the definitions for each resource. Note that the `sipserver` and `datatier` resources must each be targeted to the same servers or clusters; in Example 2–1, the resources are deployed to both the engine tier and SIP data tier cluster.

Example 2–1 Oracle WebLogic Communication Services Custom Resources

```
<custom-resource>
    <name>sipserver</name>
    <target>ORA_DATA_TIER_CLUST,ORA_ENGINE_TIER_CLUST</target>
    <descriptor-file-name>custom/sipserver.xml</descriptor-file-name>

<resource-class>com.bea.wcp.sip.management.descriptor.resource.SipServerResource</
resource-class>

<descriptor-bean-class>com.bea.wcp.sip.management.descriptor.beans.SipServerBean</
descriptor-bean-class>
</custom-resource>
<custom-resource>
    <name>datatier</name>
    <target>ORA_DATA_TIER_CLUST,ORA_ENGINE_TIER_CLUST</target>
    <descriptor-file-name>custom/datatier.xml</descriptor-file-name>

<resource-class>com.bea.wcp.sip.management.descriptor.resource.DataTierResource</r
esource-class>

<descriptor-bean-class>com.bea.wcp.sip.management.descriptor.beans.DataTierBean</d
escriptor-bean-class>
  </custom-resource>
<custom-resource>
    <name>diameter</name>
    <target>ORA_ENGINE_TIER_CLUST</target>
    <deployment-order>200</deployment-order>
    <descriptor-file-name>custom/diameter.xml</descriptor-file-name>
    <resource-class>com.bea.wcp.diameter.DiameterResource</resource-class>

<descriptor-bean-class>com.bea.wcp.diameter.management.descriptor.beans.Configurat
ionBean</descriptor-bean-class>
</custom-resource>
```

The Oracle WebLogic Communication Services custom resources utilize the basic domain resources defined in `config.xml`, such as network channels, cluster and server configuration, and Java EE resources. However, Oracle WebLogic Communication Services-specific resources are configured in separate configuration files based on functionality:

- `sipserver.xml` configures SIP container properties and general Oracle WebLogic Communication Services engine tier functionality.
- `datatier.xml` identifies servers that participate as replicas in the SIP data tier, and also defines the number and layout of SIP data tier partitions.
- `diameter.xml` configures Diameter nodes and Diameter protocol applications used in the domain.
- `approuter.xml` configures Default Application Router. For more information on configuring DAR, see *Oracle WebLogic Communication Services Installation Guide*.

Keep in mind that the domain configuration file, `config.xml`, defines all of the Managed Servers available in the domain. The `sipserver.xml`, `datatier.xml`, and `diameter.xml` configuration files included in the `sipserver` application determine the role of each server instance, such as whether they behave as SIP data tier replicas, engine tier nodes, or Diameter client nodes.

Configuration changes to SIP Servlet container properties can be applied dynamically (some SIP Servlet container properties may display a *Restart may be required* icon meaning that restart after making the change will be required) to a running server by using the Administration Console, or from the command line using the WLST utility. Configuration for SIP data tier nodes cannot be changed dynamically, so you must reboot SIP data tier servers in order to change the number of partitions or replicas.

2.2.1 Diameter Configuration

The Diameter protocol implementation is implemented as a custom resource separate from the SIP Servlet container functionality. The Diameter configuration file configures one or more Diameter protocol applications to provide Diameter node functionality. Oracle WebLogic Communication Services provides the Diameter protocol applications to support the following node types:

- *Diameter Sh* interface client node (for querying a Home Subscriber Service)
- *Diameter Rf* interface client node (for offline charging)
- *Diameter Ro* interface client node (for online charging)
- *Diameter relay node*
- *HSS simulator node* (suitable for testing and development only, not for production deployment)

The Diameter custom resource is deployed only to domains having servers that must function as Diameter client nodes or relay agents, or to servers providing HSS simulation capabilities. The actual function of the server instance depends on the configuration defined in the `diameter.xml` file.

See Section 10.2, "Steps for Configuring Diameter Client Nodes and Relay Agents" in *Configuring Network Resources* for instructions to configure the Diameter Web Application in an Oracle WebLogic Communication Services domain. See *Oracle WebLogic Communication Services Developer's Guide* for information on developing Diameter applications.

2.3 Methods and Tools for Performing Configuration Tasks

Oracle WebLogic Communication Services provides several mechanisms for changing the configuration of the SIP Servlet container:

- Section 2.3.1, "Administration Console"
- Section 2.3.2, "WebLogic Scripting Tool (WLST)"
- Section 2.3.3, "Additional Configuration Methods"

2.3.1 Administration Console

Oracle WebLogic Communication Services provides Administration Console extensions that allow you to modify and SIP Servlet container, SIP Servlet domain, and Diameter configuration properties using a graphical user interface. The Administration Console extensions for Oracle WebLogic Communication Services are

similar to the core console available in Oracle WebLogic Server 10*g* Release 3. All Oracle WebLogic Communication Services configuration and monitoring is provided via these nodes in the left pane of the console:

- SipServer—configures SIP Servlet container properties and other engine tier functionality. This extension also enables you to create new partitions, and view (but not modify) SIP data tier partitions and replicas. See Section 3.1, "Overview of SIP Container Configuration" for more information about configuring the SIP Servlet container using the Administration Console.
- Diameter—configures Diameter nodes and applications.

Note: To learn more about using Oracle WebLogic Server Administration Console, see "Getting Started with Oracle WebLogic Server Administration Console" in *Oracle Fusion Middleware Administrator's Guide*.

2.3.2 WebLogic Scripting Tool (WLST)

The WebLogic Scripting Tool (WLST) enables you to perform interactive or automated (batch) configuration operations using a command-line interface. WLST is a JMX tool that can view or manipulate the MBeans available in a running Oracle WebLogic Communication Services domain. Section 3.1, "Overview of SIP Container Configuration" provides instructions for modifying SIP Servlet container properties using WLST.

Note: To learn more about using WLST, see *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference*.

2.3.3 Additional Configuration Methods

Most Oracle WebLogic Communication Services configuration is performed using either the Administration Console or WLST. The methods described in the following sections may also be used for certain configuration tasks.

2.3.3.1 Editing Configuration Files

You may also edit `sipserver.xml`, `datatier.xml`, `diameter.xml`, and approuter.xml manually. If you edit configuration files manually, you must reboot all servers to apply the configuration changes.

2.3.3.2 Custom JMX Applications

Oracle WebLogic Communication Services properties are represented by JMX-compliant MBeans. You can therefore program JMX applications to configure SIP container properties using the appropriate Oracle WebLogic Communication Services MBeans.

The general procedure for modifying Oracle WebLogic Communication Services MBean properties using JMX is described in Section 3.3, "Configuring Container Properties Using WLST (JMX)" (WLST itself is a JMX-based application). For more information about the individual MBeans used to manage SIP container properties, see the *Oracle Fusion Middleware Communication Services Java API Reference*.

2.3.3.2.1 Setting Log Levels You can set log levels by manually editing the `logging.xml` file, by setting the `setLoggerLevel(String loggerName,`

String logLevel) MBean, or through Oracle Enterprise Manager. For more information, see Section 12.2.1, "Configuring Logging". See also *Oracle Fusion Middleware 2 Day Administration Guide.*

2.4 Starting and Stopping Servers

Oracle WebLogic Communication Services start scripts use default values for many JVM parameters that affect performance. For example, JVM garbage collection and heap size parameters may be omitted, or may use values that are acceptable only for evaluation or development purposes. In a production system, you must rigorously profile your applications with different heap size and garbage collection settings in order to realize adequate performance. See Section 8.8, "Tuning JVM Garbage Collection for Production Deployments" for suggestions about maximizing JVM performance in a production domain.

> **Caution:** When you configure a domain with multiple engine and SIP data tier servers, you must accurately synchronize all system clocks to a common time source (to within one or two milliseconds) in order for the SIP protocol stack to function properly. See Section 3.5.2, "Configuring NTP for Accurate SIP Timers" for more information.

Because a typical Oracle WebLogic Communication Services domain contains numerous engine and SIP data tier servers, with dependencies between the different server types, you should generally follow this sequence when starting up a domain:

1. **Start the Administration Server for the domain.** Start the Administration Server in order to provide the initial configuration to engine and SIP data tier servers in the domain. The Administration Server can also be used to monitor the startup/shutdown status of each Managed Server. You generally start the Administration Server by using either the `startWebLogic.cmd` script installed with the Configuration Wizard, or a custom startup script.
2. **Start SIP data tier servers in each partition.** The engine tier cannot function until servers in the SIP data tier are available to manage call state data. Although all replicas in each partition need not be available to begin processing requests, at least one replica in each configured partition must be available in order to manage the concurrent call state. All replicas should be started and available before opening the system to production network traffic.

   You generally start each SIP data tier server by using either the `startManagedWebLogic.cmd` script installed with the Configuration Wizard, or a custom startup script. `startManagedWebLogic.cmd` requires that you specify the name of the server to startup, as well as the URL of the Administration Server for the domain, as in:

   ```
   startManagedWebLogic.cmd datanode0-0 t3://adminhost:7001
   ```

3. **Start engine tier servers.** After the SIP data tier servers have started, you can start servers in the engine tier and begin processing client requests. As with SIP data tier servers, engine tier servers are generally started using the `startManagedWebLogic.cmd` script or a custom startup script.

Following the above startup sequence ensures that all Managed Servers use the latest SIP Servlet container and SIP data tier configuration. This sequence also avoids engine tier error messages that are generated when servers in the SIP data tier are unavailable.

2.5 Administration Server Best Practices

The Administration Server in a Oracle WebLogic Communication Services installation is required for configuring, deploying, and monitoring services and applications.

> **Note:** If an Administration Server fails due to a hardware, software, or network problem, only management, deployment, and monitoring operations are affected. **Managed Servers do not require the Administration Server for continuing operation; Java EE applications and SIP features running on Managed Server instances continue to function even if the Administration Server fails.**

Oracle recommends the following best practices for configuring Administration Server and Managed Server instances in your Oracle WebLogic Communication Services domain:

- Run the Administration Server instance on a dedicated machine. The Administration Server machine should have a memory capacity similar to Managed Server machines, although a single CPU is generally acceptable for administration purposes.
- Configure all Managed Server instances to use Managed Server Independence. This feature allows the Managed Servers to restart even if the Administration Server is unreachable due to a network, hardware, or software failure. See *Oracle Fusion Middleware Managing Server Startup and Shutdown for Oracle WebLogic Server* for more information.
- Configure the Node Manager utility to automatically restart all Managed Servers in the Oracle WebLogic Communication Services domain. See *Oracle Fusion Middleware Oracle WebLogic Scripting Tool* for more information.

Should an Administration Server instance or machine fail, remember that only configuration, deployment, and monitoring features are affected, but Managed Servers continue to operate and process client requests. Potential losses incurred due to an Administration Server failure include:

- Loss of in-progress management and deployment operations.
- Loss of ongoing logging functionality.
- Loss of SNMP trap generation for WebLogic Server instances (as opposed to Oracle WebLogic Communication Services instances). On Managed Servers, Oracle WebLogic Communication Services traps are generated even in the absence of the Administration Server.

To resume normal management activities, restart the failed Administration Server instance as soon as possible.

2.6 Common Configuration Tasks

General administration and maintenance of Oracle WebLogic Communication Services requires that you manage both WebLogic Server configuration properties and Oracle WebLogic Communication Services container properties. These common configuration tasks are summarized in Table 2–1.

Table 2–1 Common Oracle WebLogic Communication Services Configuration Tasks

Task	Description
Section 3.1, "Overview of SIP Container Configuration"	■ Configuring SIP Container Properties using the Administration Console ■ Using WLST to perform batch configuration
Chapter 6, "Configuring SIP Data Tier Partitions and Replicas"	■ Assigning Oracle WebLogic Communication Services instances to the SIP data tier partitions ■ Replicating call state using multiple SIP data tier instances
Chapter 4, "Managing Network Resources"	■ Configuring WebLogic Server network channels to handling SIP and HTTP traffic ■ Setting up multi-homed server hardware ■ Configuring load balancers for use with Oracle WebLogic Communication Services
Section 5.7, "Configuring Digest Authentication"	■ Configuring the LDAP Digest Authentication Provider ■ Configuring a trusted host list
Section 8.7, "Logging SIP Requests and Responses"	■ Configuring logging Servlets to record SIP requests and responses. ■ Defining log criteria for filtering logged messages ■ Maintaining Oracle WebLogic Communication Services log files

3

Configuring SIP Servlet Container Properties

The following sections describe how to configure SIP Container features in the engine tier of an Oracle WebLogic Communication Services deployment:

- Section 3.1, "Overview of SIP Container Configuration"
- Section 3.2, "Using the Administration Console to Configure Container Properties"
- Section 3.3, "Configuring Container Properties Using WLST (JMX)"
- Section 3.4, "WLST Configuration"
- Section 3.5, "Configuring Timer Processing"

3.1 Overview of SIP Container Configuration

You can configure SIP Container properties either by using a JMX utility such as the Administration Console or WebLogic Scripting Tool (WLST), or by programming a custom JMX application. Section 3.2, "Using the Administration Console to Configure Container Properties" describes how to configure container properties using the Administration Console graphical user interface.

Section 3.3, "Configuring Container Properties Using WLST (JMX)" describes how to directly access JMX MBeans to modify the container configuration. All examples use WLST to illustrate JMX access to the configuration MBeans.

3.2 Using the Administration Console to Configure Container Properties

The Administration Console included with Oracle WebLogic Communication Services enables you to configure and monitor core WebLogic Server functionality as well as the SIP Servlet container functionality provided with Oracle WebLogic Communication Services. To configure or monitor SIP Servlet features using the Administration Console, see "Getting Started with Oracle WebLogic Server Administration Console" in *Oracle Fusion Middleware Administrator's Guide*.

Table 3–1 Oracle WebLogic Communication Services ***Configuration and Monitoring Pages***

Page	SubPage	Function
Configuration	General	Configure SIP timer values, session timeout duration, default Oracle WebLogic Communication Services behavior (proxy or user agent), server header format, call state caching, DNS name resolution, timer affinity, domain aliases, rport support, and diagnostic image format.
Configuration	Application Router	Configure custom Application Router (AR) class name, configuration, or default application.
Configuration	Proxy	Configure proxy routing URIs and proxy policies.

Table 3–1 (Cont.) Oracle WebLogic Communication Services ***Configuration and Monitoring Pages***

Page	SubPage	Function
Configuration	Overload Protection	Configure the conditions for enabling and disabling automatic overload controls.
Configuration	Message Debug	Enable or disable SIP message logging on a development system.
Configuration	SIP Security	Identify trusted hosts for which authentication is not performed.
Configuration	Persistence	Configure persistence options for storing long-lived session data in an RDBMS, or for replicating long-lived session data to a remote, geographically-redundant site.
Configuration	Data Tier	View the current configuration of SIP data tier servers. You can also add, delete and configure partitions here.
Configuration	LoadBalancer Map	Configure the mapping of multiple clusters to internal virtual IP addresses during a software upgrade.
Configuration	Targets	Configure the list of servers or clusters that receive the engine tier configuration. The target server list determines which servers and/or clusters provide SIP Servlet container functionality.
Configuration	Connection Pools	Configure connection reuse pools to minimize communication overhead with a Session Border Control (SBC) function or Serving Call Session Control Function (S-CSCF).
Monitoring	General	View runtime information about messages and sessions processed in engine tier servers.
Monitoring	SIP Applications	View runtime session information for deployed SIP applications.
Monitoring	Data Tier Information	View runtime information about the current status and the work performed by servers in the SIP data tier.

3.2.1 Locking and Persisting the Configuration

In order to modify information on any of the Oracle WebLogic Communication Services configuration pages, the configuration must be locked. Locking a configuration prevents other Administrators from modifying the configuration at the same time. Locking is automatically enabled in Production domains; it can be enabled or disabled in Development domains.

To make changes:

1. Locate the Change Center in the upper left corner of the Administration Console.
2. Click **Lock & Edit** to lock the editable configuration hierarchy for the domain. This enables you to make changes using the Administration Console.
3. Make the changes you desire on the relevant page of the Console and click **Save** on each page where you make a change.
4. When you have finished making all the desired changes, click **Activate Changes** in the Change Center.

Note: Some changes you make in the Administration Console take place immediately when you activate them. Other changes require you to restart the server or module affected by the change. These latter changes are called non-dynamic changes. Non-dynamic changes are indicated in the Administration Console with a warning icon.

If an edit is made to a non-dynamic configuration setting, no edits to dynamic configuration settings will take effect until after you restart the server.

For more information on using Oracle WebLogic Server Administration Console, see *Oracle Fusion Middleware Administrator's Guide*.

3.3 Configuring Container Properties Using WLST (JMX)

The WebLogic Scripting Tool (WLST) is a utility that you can use to observe or modify JMX MBeans available on a WebLogic Server or Oracle WebLogic Communication Services instance. Full documentation for WLST is available in *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*.

Before using WLST to configure a Oracle WebLogic Communication Services domain, set you environment to add required Oracle WebLogic Communication Services classes to your classpath. Use either a domain environment script or the `setWLSEnv.sh` script located in *`MIDDLEWARE_HOME`*`/server/bin` where *`MIDDLEWARE_HOME`* is the root of your Oracle WebLogic Communication Services installation.

3.3.1 Managing Configuration Locks

Table 3–1 summarizes the WLST methods used to lock a configuration and apply changes.

Table 3–2 MBean Method Summary

Method	Description
`activate`	Writes the current configuration MBean attributes (the current SIP Servlet container configuration) to the `sipserver.xml` configuration file and applies changes to the running servers.
`cancelEdit`	Cancels an edit session, releasing the edit lock, and discarding all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.
`cd`	Navigate the hierarchy of configuration or runtime beans.
`connect`	Connect WLST to a WebLogic Server instance.
`edit`	Starts an edit session.
`save`	Writes the current configuration MBean attributes (the current SIP Servlet container configuration) to a temporary configuration file.
`set`	Set the specified attribute value for the current configuration bean.
`stopEdit`	Releases the lock obtained for modifying SIP container properties and rolls back any pending MBean changes, discarding any temporary files.

Here is an example of using the commands to modify the T1 Timer interval:

Example 3–1 Modifying T1 Timer Interval

```
connect()
edit()
cd('CustomResources/sipserver/Resource/sipserver')
set('T1TimeoutInterval',505)
activate()
```

3.3.2 Locating the Oracle WebLogic Communication Services MBeans

All SIP Servlet container configuration MBeans are located in the "serverConfig" MBean tree, accessed using the `serverConfig()` command in WLST. Within this bean tree, individual configuration MBeans can be accessed using the path:

```
CustomResources/sipserver/Resource/sipserver
```

For example, to browse the default Proxy MBean for a Oracle WebLogic Communication Services domain you would enter these WLST commands:

```
serverConfig()
cd('CustomResources/sipserver/Resource/sipserver/Proxy')
ls()
```

Runtime MBeans are accessed in the *custom* MBean tree, accessed using the `custom()` command in WLST. Runtime MBeans use the path:

```
mydomain:Location=myserver,Name=myserver,Type=mbeantype
```

Certain configuration settings, such as proxy and overload protection settings, are defined by default in `sipserver.xml`. Configuration MBeans are generated for these settings when you boot the associated server, so you can immediately browse the `Proxy` and `OverloadProtection` MBeans. Other configuration settings are not configured by default and you will need to create the associated MBeans before they can be accessed. See Section 3.4.2, "Creating and Deleting MBeans".

3.4 WLST Configuration

The following sections describe WLST scripts and commands for configuring SIP Servlet container properties.

3.4.1 Invoking WLST

To use WLST with Oracle WebLogic Communication Services, you must ensure that all Oracle WebLogic Communication Services JAR files are included in your classpath. Follow these steps:

1. Set your Oracle WebLogic Communication Services environment:

   ```
   cd MIDDLEWARE_HOME/user_projects/domains/mydomain/bin
   ./setDomainEnv.sh
   ```

2. Start WLST:

   ```
   java weblogic.WLST
   ```

3. Connect to the Administration Server for your Oracle WebLogic Communication Services domain:

```
connect('system','weblogic','t3://myadminserver:<portnumber>')
```

3.4.2 Creating and Deleting MBeans

The `SipServer` MBean represents the entire contents of the `sipserver.xml` configuration file. In addition to having several attributes for configuring SIP timers and SIP application session timeouts, `SipServer` provides helper methods to help you create or delete MBeans representing proxy settings and overload protection controls. See Table 3–1, " Oracle WebLogic Communication Services Configuration and Monitoring Pages" for a listing of other helper methods in `SipServer`; see also *Oracle Fusion Middleware Communication Services Java API Reference*.

3.5 Configuring Timer Processing

As engine tier servers add new call state data to the SIP data tier, SIP data tier instances queue and maintain the complete list of SIP protocol timers and application timers associated with each call. Engine tier servers periodically poll partitions in the SIP data tier to determine which timers have expired, given the current time. By default, multiple engine tier polls to the SIP data tier are staggered to avoid contention on the timer tables. Engine tier servers then process all expired timers using threads allocated in the `sip.timer.Default` execute queue.

3.5.1 Configuring Timer Affinity (Optional)

With the default timer processing mechanism, a given engine tier server processes all timers that are currently due to fire, regardless of whether or not that engine was involved in processing the calls associated with those timers. However, some deployment scenarios require that a timer is processed on the same engine server that last modified the call associated with that timer. One example of this scenario is a hot standby system that maintains a secondary engine that should not process any call data until another engine fails. Oracle WebLogic Communication Services enables you to configure timer affinity in such scenarios.

When you enable timer affinity, replicas request that each engine tier server periodically poll the SIP data tier for processed timers. When polling the SIP data tier, an engine processes only those timers associated with calls that were last modified by that engine, or timers for calls that have no owner.

> **Note:** When an engine tier server fails, any call states that were last modified by that engine no longer have an owner. Expired timers that have no owner are processed by the next engine server that polls the SIP data tier.

To enable timer affinity:

1. Access the Administration Console for your domain.
2. Click **Lock & Edit** (if enabled in your development environment) to obtain a configuration lock.
3. Select the SipServer node in the left pane. The right pane of the console provides two levels of tabbed pages that are used for configuring and monitoring Oracle WebLogic Communication Services.
4. Select the **Configuration > General** tab in the right pane.
5. Select *Enable Timer Affinity*.

6. Click **Save** to save your configuration changes.

Note that the Enable Timer Affinity setting is persisted in `sipserver.xml` in the element, `enable-timer-affinity`.

3.5.2 Configuring NTP for Accurate SIP Timers

In order for the SIP protocol stack to function properly, all engine and SIP data tier servers must accurately synchronize their system clocks to a common time source, to within one or two milliseconds. Large differences in system clocks cause a number of severe problems such as:

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

Oracle recommends using a Network Time Protocol (NTP) client or daemon on each Oracle WebLogic Communication Services instance and synchronizing to a common NTP server.

Caution: You must accurately synchronize server system clocks to a common time source (to within one or two milliseconds) in order for the SIP protocol stack to function properly. Because the initial T1 timer value of 500 milliseconds controls the retransmission interval for INVITE request and responses, and also sets the initial values of other timers, even small differences in system clock settings can cause improper SIP protocol behavior. For example, an engine tier server with a system clock 250 milliseconds faster than other servers will process more expired timers than other engine tier servers, will cause retransmits to begin in half the allotted time, and may force messages to timeout prematurely.

4

Managing Network Resources

The following sections describe how to configure network resources for use with Oracle WebLogic Communication Services:

4.1 Overview of Network Configuration

The default HTTP network configuration for each Oracle WebLogic Communication Services instance is determined from the Listen Address and Listen Port setting for each server. However, Oracle WebLogic Communication Services does not support the SIP protocol over HTTP. The SIP protocol is supported over the UDP and TCP transport protocols. SIPS is also supported using the TLS transport protocol.

To enable UDP, TCP, or TLS transports, you configure one or more *network channels* for a Oracle WebLogic Communication Services instance. A network channel is a configurable WebLogic Server resource that defines the attributes of a specific network connection to the server instance. Basic channel attributes include:

- The protocols supported by the connection
- The listen address (DNS name or IP address) of the connection
- The port number used by the connection
- (optional) The port number used by outgoing UDP packets
- The public listen address (load balancer address) to embed in SIP headers when the channel is used for an outbound connection.

You can assign multiple channels to a single Oracle WebLogic Communication Services instance to support multiple protocols or to utilize multiple interfaces available with multi-homed server hardware. You cannot assign the same channel to multiple server instances.

When you configure a new network channel for the SIP protocol, both the UDP and TCP transport protocols are enabled on the specified port. You cannot create a SIP

channel that supports only UDP transport or only TCP transport. When you configure a network channel for the SIPS protocol, the server uses the TLS transport protocol for the connection.

As you configure a new SIP Server domain, you will generally create multiple SIP channels for communication to each engine tier server in your system. Engine tier servers can communicate to SIP data tier replicas using the configured Listen Address attributes for the replicas. Note, however, that replicas must use unique Listen Addressees in order to communicate with one another.

Note: If you configure multiple replicas in a SIP data tier cluster, you must configure a unique Listen Address for each server (a unique DNS name or IP address). If you do not specify a unique Listen Address, the replica service binds to the default *localhost* address and multiple replicas cannot locate one another.

4.1.1 IPv4 and IPv6

If your operating system and hardware support IPv6, you can also configure Oracle WebLogic Communication Services to use IPv6 for network communication. IPv6 for SIP traffic is enabled by configuring a network channel with an IPv6 address. You must configure an IPv6 SIP channel on each engine tier server that will support IPv6 traffic. See Section 4.4.2, "Creating a New SIP or SIPS Channel" for instructions.

Note that each SIP network channel configured on an engine supports either IPv6 or IPv4 traffic. You cannot mix IPv4 and IPv6 traffic on a single channel. A single engine can be configured with both an IPv4 and IPv6 channel to support multiple, separate networks.

It is also possible for Oracle WebLogic Communication Services engine and SIP data tier nodes to communicate on IPv4 (or IPv6) while supporting the other protocol version for external SIP traffic. To configure engine and SIP data tier nodes on an IPv6 network, simply specify IPv6 listen addresses for each server instance.

See Section 4.4.2, "Creating a New SIP or SIPS Channel" for information about configuring IPv4 and IPv6 network channels.

4.2 Configuring Load Balancer Addresses

If your system uses one or more load balancers to distribute connections to the engine tier, you must configure SIP network channels to include a load balancer address as the *external listen address*. When a SIP channel has an external listen address that differs from the channel's primary listen address, Oracle WebLogic Communication Services embeds the host and port number of the external address in SIP headers such as `Response`. In this way, subsequent communication for the call is directed to the public load balancer address, rather than the local engine tier server address (which may not be accessible to external clients).

If a network channel does not have a configured external listen address, the primary listen address is embedded into SIP headers.

4.2.1 Multiple Load Balancers and Multi-homed Load Balancers

If your system uses two load balancers, you must define two channels on each engine tier server (one for each network connection to each load balancer) and assign the external listen address to the corresponding load balancer. When a particular network interface on the engine tier server is selected for outbound traffic, the network channel

associated with that Network Interface Card's (NIC) address is examined to determine the external listen address to embed in SIP headers.

If your system uses a multi-homed load balancer having two public addresses, you must also define a pair of channels to configure both public addresses. If the engine tier server has only one NIC, you must define a second, logical address on the NIC to configure a dedicated channel for the second public address. In addition, you must configure your IP routing policies to define which logical address is associated with each public load balancer address.

4.3 Enabling Domain Name Service (DNS) Support

Oracle WebLogic Communication Services supports multi-home for resolving the transport, IP address and port number of a proxy required to send a SIP message. This matches the behavior described in RFC 3263 (http://www.ietf.org/rfc/rfc3263.txt). multi-home may also used when routing responses in order to resolve the IP address and port number of a destination.

> **Note:** Because multi-home resolution is performed within the context of SIP message processing, any multi-home performance problems result in increased latency. Oracle recommends using a caching multi-home server in a production environment to minimize potential performance problems.

To configure multi-home support:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Select the *SipServer* node in the left pane of the Console.
3. Select the **Configuration > General** tab in the right pane.
4. Select **Enable multi-home Server Lookup**.
5. Click **Save** to save your changes.

When you enable multi-home lookup, the server can use multi-home to:

- Discover a proxy server's transport, IP address, and port number when a request is sent to a SIP URI.
- Resolve an IP address and/or port number during response routing, depending on the contents of the *Sent-by* field.

For proxy discovery, Oracle WebLogic Communication Services uses multi-home resolution only once per SIP transaction to determine transport, IP, and port number information. All retransmissions, ACKs, or CANCEL requests are delivered to the same address and port using the same transport. For details about how multi-home resolution takes place, see RFC 3263 (http://www.ietf.org/rfc/rfc3263.txt).

When a proxy is required to send a response message, Oracle WebLogic Communication Services uses multi-home lookup to determine the IP address and/or port number of the destination, depending on the information provided in the Sent-by field and through header.

4.4 Configuring Network Channels for SIP or SIPS

When you create a new domain using the Configuration Wizard, Oracle WebLogic Communication Services instances are configured with a default network channel supporting the SIP protocol over UDP and TCP. This default channel is configured to use Listen Port 5060, but specifies no Listen Address. Follow the instructions in Section 4.4.1, "Reconfiguring an Existing Channel" to change the default channel's listen address or listen port settings. See Section 4.4.2, "Creating a New SIP or SIPS Channel" to create a new channel resource to support additional protocols or additional network interfaces.

4.4.1 Reconfiguring an Existing Channel

You cannot change the protocol supported by an existing channel. To reconfigure an existing listen address/port combination to use a different network protocol, you must delete the existing channel and create a new channel using the instructions in Section 4.4.2, "Creating a New SIP or SIPS Channel".

To configure a channel:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane, select the **Environment > Servers** tab.
3. In the right pane, select the name of the server you want to configure.
4. Select the **Protocols > Channels** tab to display the configured channels.
5. To delete an existing channel, select it in the table and click **Delete**.
6. To reconfigure an existing channel:
   a. Select the channel's name from the channel list (for example, the default `sip` channel).
   b. Edit the *Listen Address* or *Listen Port* fields to correspond to the address of a NIC or logical address on the associated engine tier machine.

      Note: The channel must be disabled before you can modify the listen address or listen port.

   c. Edit the External Listen Address or External Listen Port fields to match the public address of a load balancer in the system.
   d. Edit the advanced channel attributes as necessary (see Section 4.4.2, "Creating a New SIP or SIPS Channel" for details.)
7. Click **Save**.

4.4.2 Creating a New SIP or SIPS Channel

To create a new SIP or SIPS channel to the configuration of a Oracle WebLogic Communication Services instance:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane, select the **Environment > Servers** tab.
3. In the right pane, select the name of the server you want to configure.

4. Select the **Protocols > Channels** tab to display the configured channels.
5. Click **New** to configure a new channel.
6. Fill in the new channel fields as follows:
   - **Name:** Enter an administrative name for this channel, such as *SIPS-Channel-eth0*.
   - **Protocol:** Select either *SIP* to support UDP and TCP transport, or *SIPS* to support TLS transport. Note that a SIP channel cannot support only UDP or only TCP transport on the configured port.
7. Click **Next**.
8. Fill in the new channel's addressing fields as follows:
   - **Listen Address:** Enter the IP address or multi-home name for this channel. On a multi-homed machine, enter the exact IP address of the interface you want to configure, or a multi-home name that maps to the exact IP address.
   - **Listen Port:** Enter the port number used to communicate through this channel. The combination of Listen Address and Listen Port must be unique across all channels configured for the server. SIP channels support both UDP and TCP transport on the configured port.
   - **External Listen Address** and **External Listen Port**: Edit these fields to match the public address of a load balancer associated with this channel. When the server selects an interface or logical address to use for outbound network traffic, Oracle WebLogic Communication Services examines the channel that was configured with the same primary *Listen Address*; if the *External Listen Address* of this channel differs, the external address is embedded into SIP message headers for further call traffic. See Section 4.2, "Configuring Load Balancer Addresses".
9. Click **Next**.
10. Optionally click **Show** to display and edit advanced channel properties, such as *connection timeout* values. Keep in mind the following restrictions and suggestions for advanced channel properties:
    - **Outbound Enabled**—This attribute cannot be unchecked, because all SIP and SIPS channels can originate network connections.
    - **HTTP Enabled for This Protocol**—This attribute cannot be selected for SIP and SIPS channels, because Oracle WebLogic Communication Services does not support HTTP transport SIP protocols.
    - **Maximum Message Size**—This attribute specifies the maximum TCP message size that the server allows on a connection from this channel. Oracle WebLogic Communication Services shuts off any connection where the messages size exceeds the configured value. The default size of 10,000,000 bytes is large. If you are concerned about preventing Denial Of Service (DOS) attacks against the server, reduce this attribute to a value that is compatible with your deployed services.
11. Click **Finish**.

4.4.3 Configuring Custom Timeout, MTU, and Other Properties

SIP channels can be further configured using one or more custom channel properties. The custom properties cannot be set using the Administration Console. Instead, you

must use a text editor to add the properties to a single, `custom-property` stanza in the channel configuration portion of the `config.xml` file for the domain.

Oracle WebLogic Communication Services provides the following custom properties that affect the transport protocol of SIP channels:

- `TcpConnectTimeoutMillis`—Specifies the amount of time Oracle WebLogic Communication Services waits before it declares a destination address (for an outbound TCP connection) as unreachable. The property is applicable only to SIP channels; Oracle WebLogic Communication Services ignores this attribute value for SIPS channels. A value of *0* disables the timeout completely. A default value of *3000 milliseconds* is used if you do not specify the custom property.
- `SctpConnectTimeoutMillis`—Specifies the amount of time Oracle WebLogic Communication Services waits before it declares a destination address (for an outbound SCTP connection) as unreachable. The property is applicable only to SCTP channels (for Diameter traffic). A value of 0 disables the timeout completely. A default value of *3000 milliseconds* is used if you do not specify the custom property. See Section 4.8.1.1, "Static Port Configuration for Outbound UDP Packets" for information about creating SCTP channels for Diameter.
- `SourcePorts`—Configures one or more static port numbers that a server uses for originating UDP packets.

  Caution: Oracle does not recommend using the *SourcePorts* custom property in most configurations because it degrades performance. Configure the property only in cases where you must specify the exact ports that Oracle WebLogic Communication Services uses to originate UDP packets.

- `Mtu`—Specifies the Maximum Transmission Unit (MTU) value for this channel. A value of -1 uses the default MTU size for the transport.
- `EnabledProtocolVersions`—Specifies the version of the SSL protocol to use with this channel when Oracle WebLogic Communication Services acts as an SSL client. When acting as an SSL client, by default the channel requires TLS V1.0 as the supported protocol. The server can be configured to use SSL V3.0 as well, if that is the highest version that the SSL peer servers support. You can set one of the following values for this property:
  - `TLS1`, the default, configures the channel to send and accept only TLS V1.0 messages. Peers must respond with a TLS V1.0 message, or the SSL connection is dropped.
  - `SSL3` configures the channel to send and accept only SSL V3.0 messages. Peers must respond with an SSL V3.0 message, or the SSL connection is dropped.
  - `ALL` supports either TLS V1.0 or SSL V3.0 messages. Peers must respond with a TLS V1.0 or SSL V3.0 message, or the SSL connection is dropped.

To configure a custom property, use a text editor to modify the `config.xml` file directly, or use a JMX client such as WLST to add the custom property. When editing `config.xml` directly, ensure that you add only one `custom-properties` element to the end of a channel's configuration stanza. Separate multiple custom properties within the same element using semicolons (;) as shown in Example 4–1.

Example 4–1 Setting Custom Properties

```
<network-access-point>
```

```
  <name>sip</name>
  <protocol>sip</protocol>
  <listen-port>5060</listen-port>
  <public-port>5060</public-port>
  <http-enabled-for-this-protocol>false</http-enabled-for-this-protocol>
  <tunneling-enabled>false</tunneling-enabled>
  <outbound-enabled>true</outbound-enabled>
  <enabled>true</enabled>
  <two-way-ssl-enabled>false</two-way-ssl-enabled>
  <client-certificate-enforced>false</client-certificate-enforced>

<custom-properties>EnabledProtocolVersions=ALL;Mtu=1000;SourcePorts=5060</custom-p
roperties>
</network-access-point>
```

4.4.4 Configuring SIP Channels for Multi-Homed Machines

If you are configuring a server that has multiple network interfaces (a *multi-homed* server), you must configure a separate network channel for each IP address used by Oracle WebLogic Communication Services. Oracle WebLogic Communication Services uses the listen address and listen port values for each channel when embedding routing information into SIP message system headers.

> **Note:** If you do not configure a channel for a particular IP address on a multi-homed machine, that IP address cannot be used when populating Via, Contact, and Record-Route headers.

4.5 Configuring TCP and TLS Channels for Diameter Support

The Oracle WebLogic Communication Services Diameter implementation supports the Diameter protocol over the TCP or TLS transport protocols. To enable incoming Diameter connections on a server, you configure a dedicated network channel using the protocol type *diameter* for TCP transport, or diameters for both TCP and TLS transport. The Diameter implementation application may automatically upgrade Diameter connections to use TLS as described in the Diameter specification (RFC 3558).

See Chapter 10, "Configuring Diameter Client Nodes and Relay Agents" for more information about configuring network channels for Diameter protocol support.

4.6 Configuring Engine Servers to Listen on Any IP Interface

To configure Oracle WebLogic Communication Services to listen for UDP traffic on any available IP interface, create a new SIP channel and specify `0.0.0.0` (or `::` for IPv6 networks) as the listen address. Note that you must still configure at least one additional channel with an explicit IP address to use for outgoing SIP messages. (For multi-homed machines, each interface used for outgoing messages must have a configured channel.)

> **Note:** If you configure a SIP channel without specifying the channel listen address, but you do configure a listen address for the server itself, then the SIP channel inherits the server listen address. In this case the SIP channel *does not* listen on IP_ANY.

Note: Using the `0.0.0.0` configuration affects only UDP traffic on Linux platforms. Oracle WebLogic Communication Services only creates TCP and HTTP listen threads corresponding to the configured host name of the server, and localhost. If multiple addresses are mapped to the host name, Oracle WebLogic Communication Services displays warning messages upon startup. To avoid this problem and listen on all addresses, specify the `::` address, which encompasses all available addresses for both IPv6 and IPv4 for HTTP and TCP traffic as well.

4.7 Configuring Unique Listen Address Attributes for SIP Data Tier Replicas

Each replica in the SIP data tier must bind to a unique Listen Address attribute (a unique multi-home name or IP address) in order to contact one another as peers. Follow these instructions for each replica to assign a unique Listen Address:

1. Access the Administration Console for the Oracle WebLogic Communication Services domain.
2. Select **Environment > Servers** from the left pane.
3. In the right pane, select the name of the server to configure.
4. Select the **Configuration > General** tab.
5. Enter a unique DNS name or IP address in the *Listen Address* field.
6. Click **Save**.

4.8 Production Network Architectures and Configuration

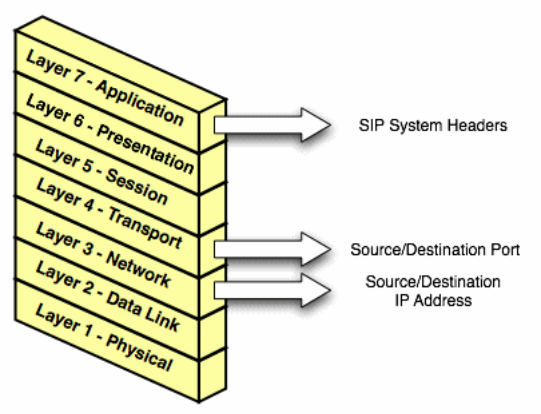
Most production (Enterprise Deployment) installations of Oracle WebLogic Communication Services contain one or more of the following characteristics:

- Multiple engine tier servers arranged in a cluster.
- Multiple network channels per engine tier server instance, in support of multiple SIP transport protocols or multiple Network Interface Cards (NICs) on multi-homed hardware.
- One or more load balancers, or a multi-homed load balancer, performing server failover and possibly Network Address Translation (NAT) for source or destination network packets.

A combination of these network elements can make it difficult to understand how elements interact with one another, and how a particular combination of elements or configuration options affects the contents of a SIP message or transport protocol datagram.

The sections that follow attempt to describe common Oracle WebLogic Communication Services network architectures and explain how servers are configured in each architecture. The sections also explain how information in SIP messages and transport datagrams is affected by each configuration. Figure 4–1 shows the typical Open Systems Interconnect (OSI) model layers that can be affected by different network configurations.

Figure 4–1 OSI Layers Affected by Oracle WebLogic Communication Services Network Configuration



Layer 3 (Network) and Layer 4 (Transport) contain the source or destination IP address and port numbers for both outgoing and incoming transport datagrams. Layer 7 (Application) may also be affected because the SIP protocol specifies that certain SIP headers include addressing information for contacting the sender of a SIP message.

4.8.1 Single-NIC Configurations with TCP and UDP Channels

In a simple network configuration for a server having a single NIC, one or more network channels may be created to support SIP messages over UDP and TCP, or SIPS over TLS. It is helpful to understand how this simple configuration affects information in the OSI model, so that you can understand how more complex configurations involving multi-homed hardware and load balancers affect the same information.

Figure 4–2 Single-NIC Network Channel Configuration

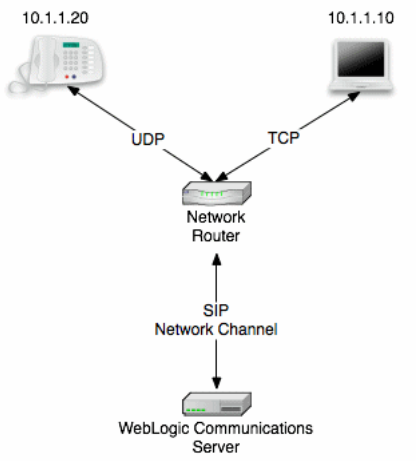


Figure 4–2 shows a single engine tier server instance with a single NIC. The server is configured with one network channel supporting SIP over UDP and TCP. (SIP channels always support both UDP and TCP transports; you cannot support only one of the two.) Figure 4–2 also shows two clients communicating with the server, one over UDP and one over TCP.

For the TCP transport, the outgoing datagram (delivered from Oracle WebLogic Communication Services to the UA) contains the following information:

- Layer 3 includes the source IP address specified by the network channel (10.1.1.10 in the example above).
- Layer 4 includes the source port number allocated by the underlying operating system.

Incoming TCP datagrams (delivered from the UA to Oracle WebLogic Communication Services) contain the following information:

- Layer 3 includes the destination IP address specified by the network channel (10.1.1.10).
- Layer 4 contains the destination port number specified by the network channel (5060).

For outgoing UDP datagrams, the OSI layer information contains the same information as with TCP transport. For incoming UDP datagrams, the OSI layer information is the same as TCP except in the case of incoming datagram Layer 4 information. For incoming UDP datagrams, Layer 4 contains either:

- The destination port number specified by the network channel (5060), or
- The ephemeral port number previously allocated by Oracle WebLogic Communication Services.

By default Oracle WebLogic Communication Services allocates ports from the ephemeral port number range of the underlying operating system for outgoing UDP datagrams. Oracle WebLogic Communication Services allows external connections to use an ephemeral point as the destination port number, in addition to the port number configured in the network channel. In other words, Oracle WebLogic Communication Services automatically listens on all ephemeral ports that the server allocates. You can optionally disable Oracle WebLogic Communication Services's use of ephemeral port numbers by specifying the following option when starting the server:

```
-Dwlss.udp.listen.on.ephemeral=false
```

You can determine Oracle WebLogic Communication Services's use of a particular ephemeral port by examining the server log file:

```
<Nov 30, 2005 12:00:00 AM PDT> <Notice> <WebLogicServer> <BEA-000202> <Thread "SIP
Message Processor (Transport UDP)" listening on port 35993.>
```

4.8.1.1 Static Port Configuration for Outbound UDP Packets

Oracle WebLogic Communication Services network channels provide a `SourcePorts` attribute that you can use to configure one or more static ports that a server uses for originating UDP packets.

> **Caution:** Oracle does not recommend using the SourcePorts custom property in most configurations because it degrades performance. Configure the property only in cases where you must specify the exact ports that Oracle WebLogic Communication Services uses to originate UDP packets.

To configure the `SourcePorts` property, use a JMX client such as WLST or directly modify a network channel configuration in `config.xml` to include the custom property. `SourcePorts` defines an array of port numbers or port number ranges. Do not include spaces in the `SourcePorts` definition; use only port numbers, hyphens ("-") to designate ranges of ports, and commas (",") to separate ranges or individual ports. See Example 4–2 for an example configuration.

Example 4–2 Static Port Configuration for Outgoing UDP Packets

```
<network-access-point>
  <name>sip</name>
  <protocol>sip</protocol>
  <listen-port>5060</listen-port>
  <public-port>5060</public-port>
  <http-enabled-for-this-protocol>false</http-enabled-for-this-protocol>
  <tunneling-enabled>false</tunneling-enabled>
  <outbound-enabled>true</outbound-enabled>
  <enabled>true</enabled>
  <two-way-ssl-enabled>false</two-way-ssl-enabled>
  <client-certificate-enforced>false</client-certificate-enforced>
  <custom-properties>SourcePorts=5060</custom-properties>
</network-access-point>
```

4.8.2 Multi-homed Server Configurations Overview

Engine tier servers in a production deployment frequently utilize multi-homed server hardware, having two or more NICs. Multi-homed hardware is typically used for one of the following purposes:

- To provide redundant network connections within the same subnet. Having multiple NICs ensures that one or more network connections are available to communicate with SIP data tier servers or the Administration Server, even if a single NIC fails.
- To support SIP communication across two or more different subnets. For example Oracle WebLogic Communication Services may be configured to proxy SIP requests from UAs in one subnet to UAs in a second subnet, when the UAs cannot directly communicate across subnets.

The configuration requirements and OSI layer information differ depending on the use of multi-homed hardware in your system. When multiple NICs are used to provide redundant connections within a subnet, servers are generally configured to listen on all available addresses (IP_ANY) as described in Section 4.8.3, "Multi-homed Servers Listening On All Addresses (IP_ANY)".

When using multiple NICs to support different subnets, you must configure multiple network on the server for each different NIC as described in Section 4.8.4, "Multi-homed Servers Listening on Multiple Subnets".

4.8.3 Multi-homed Servers Listening On All Addresses (IP_ANY)

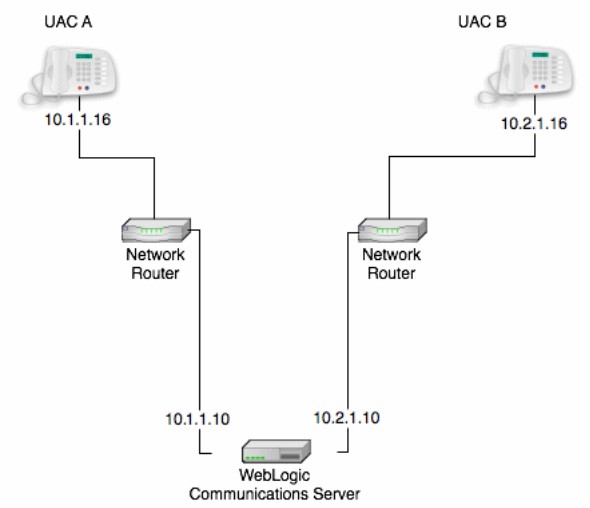
The simplest multi-home configuration enables a Oracle WebLogic Communication Services instance to listen on all available NICs (physical NICs as well as logical NICs), sometimes described as IP_ANY. To accomplish this, you configure a single network channel and specify a channel listen address of 0.0.0.0.

Note that you must configure the 0.0.0.0 address directly on the server's network channel. If you specify no IP address in the channel, the channel inherits the listen address configured for the server instance itself. See Section 4.6, "Configuring Engine Servers to Listen on Any IP Interface".

4.8.4 Multi-homed Servers Listening on Multiple Subnets

Multiple NICs can also be used in engine tier servers to listen on multiple subnets. The most common configuration uses Oracle WebLogic Communication Services to proxy SIP traffic from one subnet to another where no direct access between subnets is permitted. Figure 4–3 shows this configuration.

Figure 4–3 Multi-homed Configuration for Proxying between Subnets



To configure the Oracle WebLogic Communication Services instance in Figure 4–3 you must define a separate network channel for each NIC used on the server machine. Example 4–3 shows the `config.xml` entries that define channels for the sample configuration.

Example 4–3 Sample Network Channel Configuration for NICs on Multiple Subnets

```
<NetworkAccessPoint ListenAddress="10.1.1.10" ListenPort="5060" Name="sipchannelA"
Protocol="sip"/>
<NetworkAccessPoint ListenAddress="10.2.1.10" ListenPort="5060" Name="sipchannelB"
Protocol="sip"/>
```

4.8.4.1 Understanding the Route Resolver

When Oracle WebLogic Communication Services is configured to listen on multiple subsets, a feature called the *route resolver* is responsible for the following activities:

- Populating OSI Layer 7 information (SIP system headers such as Via, Contact, and so forth) with the correct address.
- Populating OSI Layer 3 information with the correct source IP address.

For example, in the configuration shown in Figure 4–3, Oracle WebLogic Communication Services must add the correct subnet address to SIP system headers and transport datagrams in order for each UA to continue processing SIP transactions. If the wrong subnet is used, replies cannot be delivered because neither UA can directly access the other UA's subnet.

The route resolver works by determining which NIC the operating system will use to send a datagram to a given destination, and then examining the network channel that is associated with that NIC. It then uses the address configured in the selected network channel to populate SIP headers and Layer 3 address information.

For example, in the configuration shown in Figure 4–3, an INVITE message sent from Oracle WebLogic Communication Services to UAC B would have a destination address of 10.2.1.16. The operating system would transmit this message using NIC B, which is configured for the corresponding subnet. The route resolver associates NIC B with the configured `sipchannelB` and embeds the channel's IP address (10.2.1.10) in the VIA header of the SIP message. UAC B then uses the VIA header to direct

subsequent messages to the server using the correct IP address. A similar process is used for UAC A, to ensure that messages are delivered only on the correct subnet.

4.8.4.2 IP Aliasing with Multi-homed Hardware

IP aliasing assigns multiple logical IP addresses to a single NIC, and is configured in the underlying server operating system. If you configure IP aliasing and all logical IP addresses are within the same subnet, you can simply configure Oracle WebLogic Communication Services to listen on all addresses as described in Section 4.8.3, "Multi-homed Servers Listening On All Addresses (IP_ANY)".

If you configure IP aliasing to create multiple logical IP addresses on different subnets, you must configure a separate network channel for each logical IP address. In this configuration, Oracle WebLogic Communication Services treats all logical addresses as separate physical interfaces (NICs) and uses the route resolver to populate OSI Layer 4 and Layer 7 information based on the configured channel.

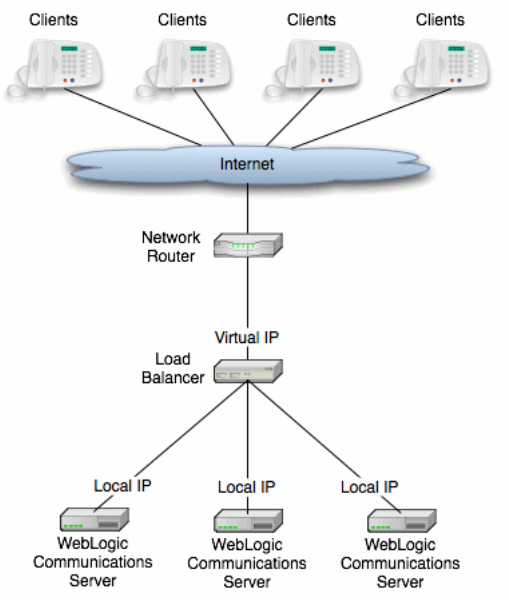
4.8.5 Load Balancer Configurations

In addition to providing failover capabilities and distributing the client load across multiple servers, a load balancer is also an important tool for configuring the network information transmitted between clients and servers. The sections that follow describe common load balancer configurations used with Oracle WebLogic Communication Services.

4.8.5.1 Single Load Balancer Configuration

The most common load balancer configuration utilizes a single load balancer that gates access to a cluster of engine tier servers, as shown in Figure 4–4.

Figure 4–4 SIngle Load Balancer Configuration



To configure Oracle WebLogic Communication Services for use with a single load balancer as in Figure 4–4, configure one or more network channels for each server, and configure the public address of each channel with the Virtual IP address of the load balancer. In this configuration, Oracle WebLogic Communication Services embeds the load balancer IP address in SIP message system headers to ensure that clients can

reach the cluster for subsequent replies. Chapter 4, "Managing Network Resources" presents detailed steps for configuring network channels with load balancer addresses.

> **Note:** Although some load balancing switches can automatically re-route all SIP messages in a given call to the same engine tier server, this functionality is not required with Oracle WebLogic Communication Services installations. See Section 15.7, "Alternate Configurations" for more information.

4.8.5.2 Multiple Load Balancers and Multi-homed Load Balancers

Multiple load balancers (or a multi-homed load balancer) can be configured to provide several virtual IP addresses for a single Oracle WebLogic Communication Services cluster. To configure Oracle WebLogic Communication Services for use with a multi-homed load balancer, you create a dedicated network channel for each load balancer or local server NIC, and set the channel's public address to the virtual IP address of the appropriate load balancer. In this configuration, the route resolver associates a configured channel with the NIC used for originating SIP messages. The public address of the selected channel is then used for populating SIP system messages. See Section 4.8.4.1, "Understanding the Route Resolver".

4.8.5.3 Network Address Translation Options

In the most common case, a load balancer is configured using destination NAT to provide a public IP address that clients use for communicating with one or more internal (private) Oracle WebLogic Communication Services addresses. Load balancers may also be configured using source NAT, which modifies the Layer 3 address information originating from a private address to match the virtual IP address of the load balancer itself.

With the default route resolver behavior, an Oracle WebLogic Communication Services engine originates UDP packets having a source IP address that matches the address of a local NIC (the private address). This can be problematic for applications that try to respond directly to the Layer 3 address embedded in the transport packet, because the local server address may not be publicly accessible. If your applications exhibit this problem, Oracle recommends that you configure the load balancer to perform source NAT to change the transport Layer 3 address to a publicly-accessible virtual IP address.

4.8.5.3.1 IP Masquerading Alternative to Source NAT If you choose not to enable source NAT on your load balancer, Oracle WebLogic Communication Services provides limited IP masquerading functionality. To use this functionality, configure a logical address on each engine tier server using the public IP address of the load balancer for the cluster. (Duplicate the same logical IP address on each engine tier server machine). When a local server interface matches the IP address of a configured load balancer (defined in the public address of a network channel), Oracle WebLogic Communication Services uses that interface to originate SIP UDP messages, and the Layer 3 address contains a public address.

> **Caution:** Using the Oracle WebLogic Communication Services IP masquerading functionality can lead to network instability, because it requires duplicate IP addresses on multiple servers. Production deployments must use a load balancer configured for source NAT, rather than IP masquerading, to ensure reliable network performance.

You can disable IP masquerading functionality by using the startup option:

```
-Dwlss.udp.lb.masquerade=false
```

4.9 Example Network Configuration

Oracle WebLogic Communication Services is compatible with load balancers that are not SIP-aware, meaning that they do not consider existing SIP dialogues when routing requests to servers. This document demonstrates load balancer and Oracle WebLogic Communication Services configuration, as well as SIP and Network Address Translation (NAT) interactions in various configurations.

The following sections describe a sample network configuration for Oracle WebLogic Communication Services using a non-SIP-aware load balancer:

For more information about implementation-dependent issues surrounding NAT see the IETF document, *NAT Behavioral Requirements for Unicast UDP* at http://tools.ietf.org/wg/behave/draft-ietf-behave-nat-udp/.

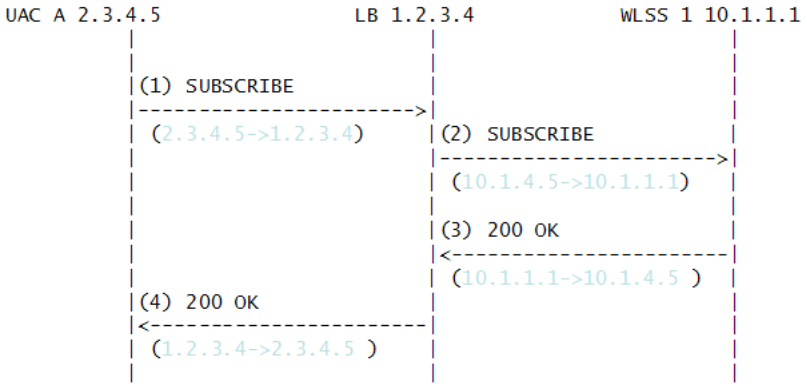
4.9.1 Example Network Topology

Figure 4–5 shows the sample network topology described in this section. An Oracle WebLogic Communication Services cluster, consisting of engines WLSS 1 and WLSS 2, is configured on private IP network 10.1/16 (an internal 16-bit subnet). The cluster's public IP address is 1.2.3.4, which is the virtual IP address configured on the load balancer.

The User Agent, UAC A, with IP address 2.3.4.5 never sees the internal IP addresses configured for the Oracle WebLogic Communication Services cluster. Instead, it sends requests to, and receives responses from 1.2.3.4.

The sections that follow discuss configuring the Oracle WebLogic Communication Services cluster and load balancer for this example system.

Figure 4–5 Example Network Topology



4.9.2 Oracle WebLogic Communication Services Configuration

The Oracle WebLogic Communication Services cluster configuration specifies the public address as 1.2.3.4, and the public port as 5060 (see Section 4.2, "Configuring Load Balancer Addresses") for each engine. The default route on both Oracle WebLogic Communication Services engines points to the load balancer's 10.1/16 network interface: 10.1.3.4. The Oracle WebLogic Communication Services (servers WLSS 1 and WLSS 2) routing table is shown in Example 4–4.

Example 4–4 Oracle WebLogic Communication Services Routing Table

```
$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.0           *            255.255.0.0     U     0      0        0 eth0
default         10.1.3.4        0.0.0.0         UG    0      0
```

4.9.3 Load Balancer Configuration

The load balancer is configured with a virtual IP address of 1.2.3.4, and two real servers, WLSS 1 and WLSS 2, having addresses 10.1.1.1 and 10.1.1.2, respectively. The load balancer also has an internal IP address of 10.1.3.4 configured on the 10.1/16 network. The UAC address, 2.3.4.5, is reachable from the load balancer by static route configuration on the load balancer. The load balancer routing table is shown in Example 4–5.

Example 4–5 Load balancer Routing Table

```
$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.0           *            255.255.0.0     U     0      0        0 eth1
1.2.0.0            *            255.255.0.0     U     0      0
```

Because the SIP protocol specification (RFC 3261) dictates the destination IP address and UDP port numbers that user agents must use when sending requests or responses, the NAT configuration of the load balancer must be done in a way that does not violate RFC 3261 requirements. Three setup options can be used to accomplish this configuration:

- Section 4.9.3.1, "NAT-based configuration"
- Section 4.9.3.2, "maddr-Based Configuration"
- Section 4.9.3.3, "rport-Based Configuration"

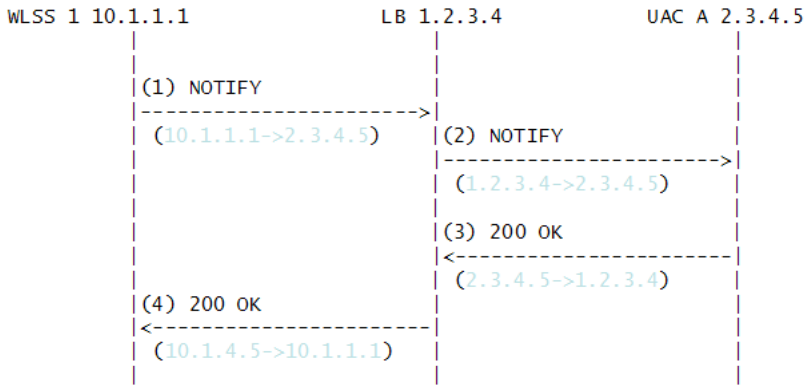
The sections that follow describe each approach.

4.9.3.1 NAT-based configuration

The default UDP NAT behavior for load balancers is to perform destination IP address translation in the public > private network direction, and source IP address translation in the private > public network direction. This means setting up destination address translation in the UAC > Oracle WebLogic Communication Services (2.3.4.5 > 1.2.3.4) direction without source address translation, and source address translation in the Oracle WebLogic Communication Services > UAC (10.1/16 > 2.3.4.5) direction without destination address translation.

Figure 4–6 illustrates the UDP packet flow for a SUBSCRIBE/200OK transaction.

Figure 4–6 SUBSCRIBE Sequence



Note that the source and destination IP addresses of the UDP packets are shown in blue. In the UAC > Oracle WebLogic Communication Services direction, the load balancer translates the destination IP address but not the source IP address. In the Oracle WebLogic Communication Services > UAC direction, the load balancer translates the source IP address but not the destination IP address.

The complete message trace (including IP and UDP headers, as well as the SIP payload) for the sequence from Figure 4–6 is shown in Example 4–6 below.

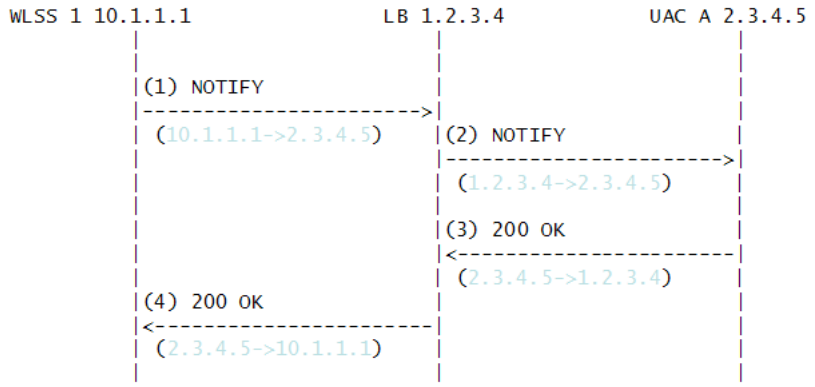
Example 4–6 Complete SUBSCRIBE Message Trace

```
No.     Time        Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request: SUBSCRIBE
sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      2 2.426250    2.3.4.5           10.1.1.1          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
```

```
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      3 3.430903    10.1.1.1               2.3.4.5           SIP      Status: 200
OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 SUBSCRIBE
        Call-ID: 1-25923@2.3.4.5
```

If Oracle WebLogic Communication Services subsequently sends a NOTIFY request to the UAC, the sequence shown in Figure 4–7 takes place:

Figure 4–7 NOTIFY Sequence



As in the previous sequence, the IP address translation takes place in the Oracle WebLogic Communication Services > UAC direction for the source IP address, and UAC > Oracle WebLogic Communication Services direction for the destination IP address.

Note that this setup does not require the load balancer to maintain session state information or to be SIP-aware. The complete message trace from Figure 4–7 is shown in Example 4–7 below.

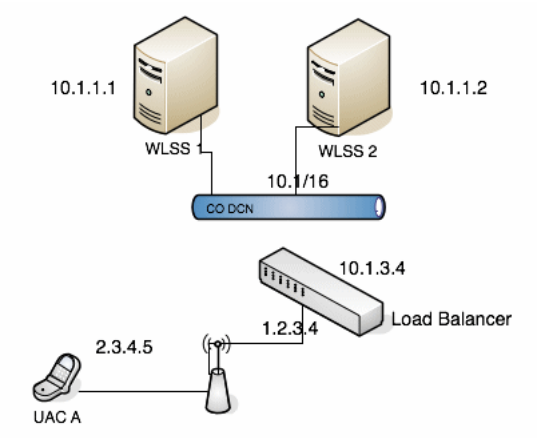
Example 4–7 Complete NOTIFY Message Trace

```
No.     Time        Source                Destination           Protocol Info
      1 5.430952    10.1.1.1               2.3.4.5           SIP      Request:
NOTIFY sip:sipp@2.3.4.5:9999

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
```

```
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70

No.     Time        Source                Destination           Protocol Info
      2 6.430952    1.2.3.4          2.3.4.5           SIP      Request: NOTIFY
sip:sipp@2.3.4.5:9999

Internet Protocol, Src: 1.2.3.4 (1.2.3.4), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70


No.     Time        Source                Destination           Protocol Info
      3 7.431367    2.3.4.5           1.2.3.4          SIP      Status: 200 OK

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        To: sipp <sip:sipp@2.3.4.5>;tag=1;tag=1
        Call-ID: 1-25923@2.3.4.5
        CSeq: 1 NOTIFY
        Contact: <sip:2.3.4.5:9999;transport=UDP>
```

Caution: If NAT is performed on both the source (SNAT) and destination IP addresses, the configuration does not work because the load balancer usually relies on a specific destination port number value to be sent in responses to requests. That port number value is dictated by RFC 3261, and must come from the Via header, which presents a conflict with load balancer's NAT requirements. RFC 3261 requires that responses to SIP requests be sent to the IP address used to send the request (unless maddr is present in the Via, as described in Section 4.9.3.2, "maddr-Based Configuration"). Consequently, in Figure 4–8 below, Step 3, Oracle WebLogic Communication Services sends a 200 OK response to the load balancer internal IP address (10.1.3.4) and port 5060. That response is then dropped.

Figure 4–8 Source and Destination NAT



The complete message trace from Figure 4–8 is show in Example 4–8 below.

Example 4–8 Complete Failing SUBSCRIBE Message Trace

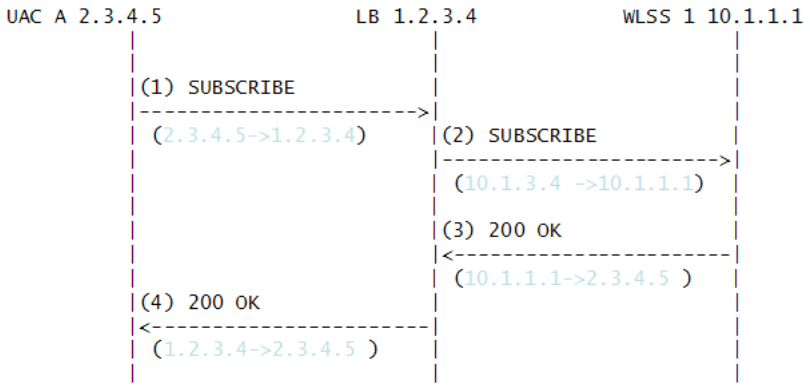
```
No.     Time        Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request: SUBSCRIBE
sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      2 2.426250    10.1.3.4          10.1.1.1          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060
```

```
Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      3 3.430903    10.1.1.1               10.1.3.4           SIP      Status: 200
OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.3.4 (10.1.3.4)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
```

4.9.3.2 maddr-Based Configuration

When the maddr parameter is present in the Via header, the response is sent to the IP address specified in the maddr rather than to the received IP address (even when SNAT is enabled). In the example below, the UAC specifies a maddr set to 2.3.4.5 in the Via header. Consequently the response from the SIP server makes it to the UAC.

Figure 4–9 maddr Sequence



The complete message trace from Figure 4–9 is shown in Example 4–9 below.

Example 4–9 Complete maddr Message Trace

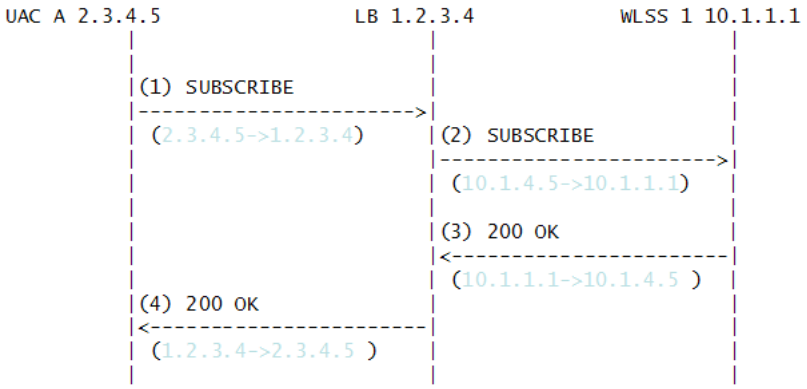
```
No.     Time        Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request: SUBSCRIBE
sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
```

```
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;maddr=2.3.4.5;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      2 2.426250    10.1.3.4           10.1.1.1          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;maddr=2.3.4.5;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      3 3.430903    10.1.1.1               2.3.4.5           SIP      Status: 200
OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
```

4.9.3.3 rport-Based Configuration

RFC 3581 improves SIP and NAT interactions by allowing the client to request that the server send responses to a UDP port number from the request rather than from the Via. In order for both SUBSCRIBE and NOTIFY to work correctly, both the UAC as well as Oracle WebLogic Communication Services must support RFC 3581. Figure 4–10 illustrates the SUBSCRIBE flow.

Figure 4–10 rport SUBSCRIBE Sequence



The complete message trace from Figure 4–10 is shown in Example 4–10 below.

Example 4–10 Complete Message Trace for rport SUBSCRIBE

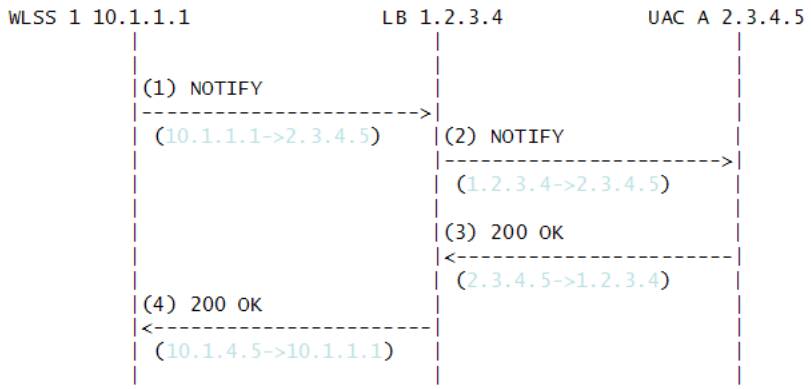
```
No.     Time        Source                Destination           Protocol Info
      1 1.425250    2.3.4.5           1.2.3.4          SIP      Request: SUBSCRIBE
sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;rport;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0

No.     Time        Source                Destination           Protocol Info
      2 2.426250    10.1.3.4          10.1.1.1          SIP      Request:
SUBSCRIBE sip:subscribe@1.2.3.4:5060

Internet Protocol, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: sip (5060)
Session Initiation Protocol
    Request-Line: SUBSCRIBE sip:subscribe@1.2.3.4:5060 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 2.3.4.5:9999;rport;branch=1
        From: sipp <sip:sipp@2.3.4.5>;tag=1
        To: sut <sip:subscribe@1.2.3.4:5060>
        Call-ID: 1-25923@2.3.4.5
        Cseq: 1 SUBSCRIBE
        Contact: sip:sipp@2.3.4.5:9999
        Max-Forwards: 70
        Event: ua-profile
        Expires: 10
        Content-Length: 0
```

```
No.     Time        Source                Destination           Protocol Info
      3 3.430903    10.1.1.1               10.1.3.4           SIP      Status: 200
OK

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 10.1.3.4 (10.1.3.4)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 2222 (2222)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        To: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 SUBSCRIBE
        Call-ID: 1-25923@2.3.4.5
```

Figure 4–11 illustrates the NOTIFY flow.

Note that while source address NAT is enabled for both directions (**UAS > Oracle WebLogic Communication Services** and **Oracle WebLogic Communication Services > UA**), the load balancer can correctly identify the destination address in Step 3 by relying on receiving responses on the same port number as the one used to send requests. This implies that the load balancer maintains state.

Figure 4–11 rport NOTIFY Sequence



The complete message trace from Figure 4–11 is shown in Example 4–11 below.

Example 4–11 Complete Message Trace for rport NOTIFY

```
No.     Time        Source                Destination           Protocol Info
      1 5.430952    10.1.1.1               2.3.4.5            SIP      Request:
NOTIFY sip:sipp@2.3.4.5:9999

Internet Protocol, Src: 10.1.1.1 (10.1.1.1), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 42316 (42316), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
        Via: SIP/2.0/UDP
```

```
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
;rport
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70

No.     Time        Source                Destination           Protocol Info
      2 6.430952    1.2.3.4          2.3.4.5           SIP      Request: NOTIFY
sip:sipp@2.3.4.5:9999

Internet Protocol, Src: 1.2.3.4 (1.2.3.4), Dst: 2.3.4.5 (2.3.4.5)
User Datagram Protocol, Src Port: 2222 (2222), Dst Port: 9999 (9999)
Session Initiation Protocol
    Request-Line: NOTIFY sip:sipp@2.3.4.5:9999 SIP/2.0
    Message Header
        To: sipp <sip:sipp@2.3.4.5>;tag=1
        Content-Length: 0
        Contact:
<sip:app-12eomtm5h5f77@1.2.3.4:5060;transport=udp;wlsscid=1ae4479ac6ff71>
        CSeq: 1 NOTIFY
        Call-ID: 1-25923@2.3.4.5
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
;rport
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        Max-Forwards: 70


No.     Time        Source                Destination           Protocol Info
      3 7.431367    2.3.4.5           1.2.3.4          SIP      Status: 200 OK

Internet Protocol, Src: 2.3.4.5 (2.3.4.5), Dst: 1.2.3.4 (1.2.3.4)
User Datagram Protocol, Src Port: 9999 (9999), Dst Port: (2222)
Session Initiation Protocol
    Status-Line: SIP/2.0 200 OK
    Message Header
        Via: SIP/2.0/UDP
1.2.3.4:5060;wlsscid=1ae4479ac6ff71;branch=z9hG4bKc5e4c3b4c22be517133ab749adeece4e
;rport
        From: sut <sip:subscribe@1.2.3.4:5060>;tag=82722c03
        To: sipp <sip:sipp@2.3.4.5>;tag=1;tag=1
        Call-ID: 1-25923@2.3.4.5
        CSeq: 1 NOTIFY
        Contact: <sip:2.3.4.5:9999;transport=UDP
```

5 Administering Security Features

The following sections provide an overview of Oracle WebLogic Communication Services security:

- Section 5.1, "Authentication for SIP Servlets"
- Section 5.2, "Overriding Authentication with Trusted Hosts"
- Section 5.3, "Identity Assertion Support"
- Section 5.4, "Role Assignment for SIP Servlet Declarative Security"
- Section 5.5, "Security Event Auditing"
- Section 5.6, "Common Security Configuration Tasks"

5.1 Authentication for SIP Servlets

Oracle WebLogic Communication Services users must be authenticated when they request access to a protected resource, such as a protected method within a deployed SIP Servlet. Oracle WebLogic Communication Services enables you to implement user authentication for SIP Servlets using any of the following techniques:

- **DIGEST authentication** uses a simple challenge-response mechanism to verify the identity of a user over SIP. This technique is described in Section 5.7, "Configuring Digest Authentication." To authenticate over HTTP, application developers must provide their own implementations.
- **CLIENT-CERT authentication** uses an X509 certificate chain passed to the SIP application to authenticate a user. The X509 certificate chain can be provided in a number of different ways. In the most common case, two-way SSL handshake is performed before transmitting the chain to ensure secure communication between the client and server. CLIENT-CERT authentication is described fully in Section 5.8, "Configuring Client-Cert Authentication."

Different SIP Servlets deployed on Oracle WebLogic Communication Services can use different authentication mechanisms as necessary. The required authentication mechanism is specified in the `auth-method` element of the SIP Servlet's `sip.xml` deployment descriptor. The deployment descriptor may also define which resources are to be protected, listing specific role names that are required for access. The SIP Servlet v1.1 specification introduces the ability to specify the realm name and identity assertion mechanism required or supported by an application. See the SIP Servlet v1.1 specification for information about defining the Servlet authentication and identity assertion mechanism.

5.1.1 Authentication Providers

Oracle WebLogic Communication Services authentication services are implemented using one or more authentication providers. An authentication provider performs the work of proving the identity of a user or system process, and then transmitting the identity information to other components of the system.

You can configure and use multiple authentication providers to use different authentication methods, or to work together to provide authentication. For example, when using Digest authentication you typically configure both a Digest Identity Asserter provider to assert the validity of a digest, and a second LDAP or RDBMS authentication provider that determines the group membership of a validated user.

When linking multiple authentication providers, you must specify the order in which providers are used to evaluate a given user, and also specify how much control each provider has over the authentication process. Each provider can contribute a "vote" that specifies whether or not the provider feels a given user is valid. The provider's control flag indicates how the provider's vote is used in the authentication process.

For more information about configuring providers, see either Section 5.7, "Configuring Digest Authentication" or Section 5.8, "Configuring Client-Cert Authentication."

5.2 Overriding Authentication with Trusted Hosts

Oracle WebLogic Communication Services also enables you to designate trusted hosts for your system. Trusted hosts are hosts for which Oracle WebLogic Communication Services performs no authentication. If the server receives a SIP message having a destination address that matches a configured trusted host name, the message is delivered without Authentication. See *Oracle Fusion Middleware Security Guide* for more information.

5.3 Identity Assertion Support

Oracle WebLogic Communication Services supports the `P-Asserted-Identity` SIP header as described in RFC 3325. This functionality automatically logs in using credentials specified in the `P-Asserted-Identity` header when they are received form a trusted host. When combined with the `privacy` header, `P-Asserted-Identity` also determines whether the message can be forwarded to trusted and non-trusted hosts.

Oracle WebLogic Communication Services also supports identity assertion using the `Identity` and `Identity-Info` headers as described in RFC 4474.

Both identity assertion mechanisms require that you configure an appropriate security provider with Oracle WebLogic Communication Services. See Section 5.9, "Configuring SIP Servlet Identity Assertion Mechanisms" for more information.

5.4 Role Assignment for SIP Servlet Declarative Security

The SIP Servlet API specification defines a set of deployment descriptor elements that can be used for providing declarative and programmatic security for SIP Servlets. The primary method for declaring security constraints is to define one or more `security-constraint` elements and role definitions in the `sip.xml` deployment descriptor. Oracle WebLogic Communication Services adds additional deployment descriptor elements to help developers easily map SIP Servlet roles to actual principals and/or roles configured in the SIP Servlet container.

5.5 Security Event Auditing

Oracle WebLogic Communication Services includes an auditing provider that you can configure to monitor authentication events in the security realm. See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information.

5.6 Common Security Configuration Tasks

Table 5–1 lists Oracle WebLogic Communication Services configuration tasks and provides links to additional information.

Table 5–1 Security Configuration Tasks

Task	Description
Section 5.7, "Configuring Digest Authentication"	▪ Understanding the Digest identity assertion providers ▪ Configuring LDAP Digest authentication ▪ Configuring Digest authentication with an RDBMS
Section 5.8, "Configuring Client-Cert Authentication"	▪ Understanding client-cert authentication solutions ▪ Delivering X509 certificates over 2-way SSL ▪ Developing a Perimeter authentication solution ▪ Using the Oracle WebLogic Communication Services `WL_Client_Cert` header to deliver X509 certificates
Section 5.9, "Configuring SIP Servlet Identity Assertion Mechanisms"	▪ Understand forwarding rules for SIP messages having the `P-Asserted-Identity` header ▪ Configuring `P-Asserted-Identity` providers

5.7 Configuring Digest Authentication

The following sections provide a basic overview of Digest authentication, and describe Digest authentication support and configuration in Oracle WebLogic Communication Services.

5.7.1 What Is Digest Authentication?

Digest authentication is a simple challenge-response mechanism used to authenticate a user over SIP or HTTP. Digest authentication is fully described in RFC 2617.

When using Digest authentication, if a client makes an un-authenticated request for a protected server resource, the server challenges the client using a nonce value. The client uses a requested algorithm (MD5 by default) to generate an encrypted response—a Digest—that includes a username, apssword, and realm.

The server verifies the client Digest by recreating the Digest value and comparing it with the client's Digest. To recreate the Digest value the server requires a hash of the "A1" value (see RFC 2617) that includes, at minimum, the nonce, username, password and realm name. The server either recreates the hash of the A1 value using a stored clear-text password for the user, or by obtaining a precalculated hash value. Either the clear-text password or precalculated hash value can be stored in an LDAP directory or accessed from an RDBMS using JDBC. The server then uses the hash of the A1 value to recreate the Digest and compare it to the client's Digest to verify the user's identity.

Digest authentication provides secure authorization over HTTP because the clear text password is never transmitted between the client and server. The use of nonce values in the client challenge also ensures that Digest authentication is resistant to replay

attacks. See Figure 5–1 for a more detailed explanation of the challenge-response mechanism for a typical request.

5.7.2 Digest Authentication Support in Oracle WebLogic Communication Services

Oracle WebLogic Communication Services includes LDAP Digest Identity Asserter security providers for asserting the validity of a client's Digest using LDAP or an RDBMS. A separate authorization provider is required to complete the authentication process (see Section 5.7.4.3, "Configure an Authenticator Provider").

The Digest Identity Asserter only verifies a user's credentials using the client Digest. After the Digest is verified, the configured authorization provider completes the authentication process by checking for the existence of the user (by username) and also populating group membership for the resulting `javax.security.auth.Subject`.

The Digest Identity Asserter provider requires that user credentials be stored in an LDAP server or RDBMS in one of the following ways:

- **Unencrypted (clear text) passwords.** The simplest configuration stores users' unencrypted passwords in a store. If you choose this method, Oracle recommends using an SSL connection to the LDAP store or database to reduce the risk of exposing clear text passwords in server-side network traffic. Some LDAP stores do not support storing unencrypted passwords by default; in this case you must create or use a dedicated credential attribute on the LDAP server for storing the password. See Section 5.7.4.1, "Configure the LDAP Server or RDBMS".
- **Reverse-Encrypted Passwords.** Oracle WebLogic Communication Services provides a utility to help you compute the Encryption Key, Encryption Init Vector, and Encrypted Passwords values used when you configure the Digest Authorization Identity Asserter provider.
- **A pre-calculated hash of each password, username, and realm.** If storing unencrypted or reverse-encrypted passwords is unacceptable, you can instead store a pre-calculated hash value of the username, security-realm, and password in a new or existing attribute in LDAP or an RDBMS. The Digest Identity Asserter then retrieves only the hash value for comparison to the client-generated hash in the Digest. Storing pre-calculated hash values provides additional security.

The LDAP Digest Identity Asserter is compatible with any LDAP provider that permits storage of a clear text password or pre-calculated hash value.

Note: You cannot change the schema for the built-in LDAP store to add a dedicated field for storing clear text passwords or pre-calculated hash values. However, you can use the predefined "description" field to store password information for testing or demonstration purposes.

If you do not use the DefaultAuthenticator provider for authentication decisions, you must make DefaultAuthenticator an optional provider (ControlFlag="SUFFICIENT" or lower) before you can use Digest authentication. This will generally be the required configuration in production installations where a separate LDAP store is used to maintain clear text or hashed password information.

Figure 5–1 Digest Authentication in Oracle WebLogic Communication Services

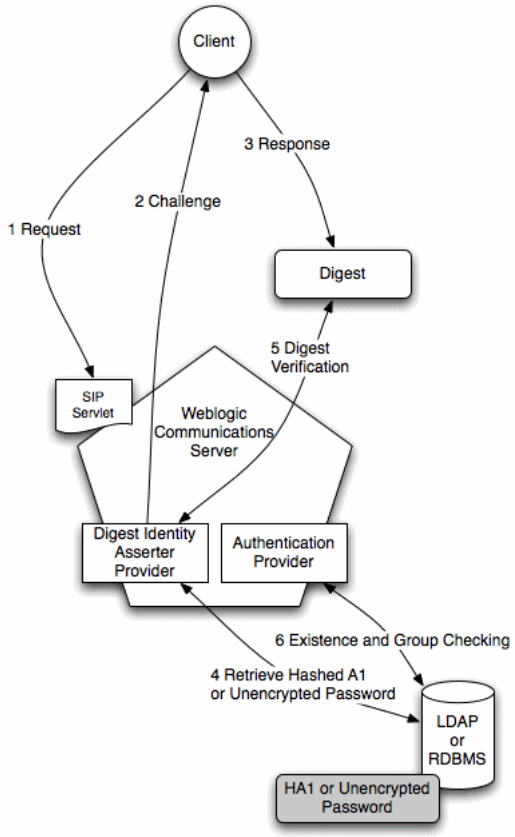


Figure 5–1 shows the basic architecture and use of an Identity Asserter provider for a typical client request:

1. The client makes an unauthorized request for a protected application resource. (SIP Servlet resources can be protected by specifying security constraints in the `sip-xml` deployment descriptor.
2. The Digest Identity Asserter provider generates a challenge string consisting of the nonce value, realm name, and encryption algorithm (either MD5 or MD5-sess). The SIP container delivers the challenge string to the client.

> **Note:** The Digest Identity Asserter maintains a cache of used nonces and timestamps for a specified period of time. All requests with a timestamp older than the specified timestamp are rejected, as well as any requests that use the same timestamp/nonce pair as the most recent timestamp/nonce pair still in the cache.

3. The client uses the encryption algorithm to create a Digest consisting of the username, password, real name, nonce, SIP method, request URI, and other information described in RFC 2617.
4. The Digest Identity Asserter verifies the client Digest by recreating the Digest value using a hash of the A1 value, nonce, SIP method, and other information. To obtain a hash of the A1 value, the Identity Asserter either generates HA1 by retrieving a clear-text password from the store, or the Identity Asserter retrieves the pre-calculated HA1 from the store.

5. The generated Digest string is compared to the client's Digest to verify the user's identity.
6. If the user's identity is verified, an authentication provider then determines if the user exists and if it does, the authentication provider populates the `javax.security.auth.Subject` with the configured group information. This step completes the authentication process.

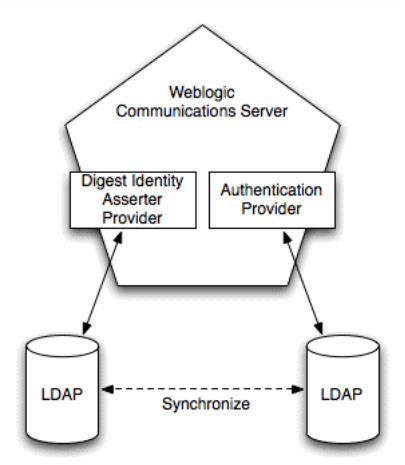
> **Note:** If you do not require user existence checking or group population, you can use the special "no-op" Identity Assertion Authenticator to avoid an extra connection to the LDAP Server; see Section 5.7.4.3, "Configure an Authenticator Provider" for more information.

After authentication is complete, the SIP Servlet container performs an authorization check for the logged in `javax.security.auth.Subject` against the declarative security-constraints defined in the Servlet's `sip.xml` deployment descriptor.

The LDAP Digest Identity Asserter and the configured Authentication provider can either use the same LDAP store or different stores.

> **Note:** If you use multiple LDAP stores, you must also create some infrastructure to keep both stores synchronized in response to adding, removing, or changing user credential changes, as shown in Figure 5–2. Maintaining LDAP stores in this manner is beyond the scope of this documentation.

Figure 5–2 Multiple LDAP Servers



5.7.3 Prerequisites for Configuring LDAP Digest Authentication

In order to configure Digest authentication you must understand the basics of LDAP servers and LDAP administration. You must also understand the requirements and restrictions of your selected LDAP server implementation, and have privileges to modify the LDAP configuration as well as the Oracle WebLogic Communication Services configuration.

Table 5–1 summarizes all of the information you will need in order to fully configure your LDAP server for Digest authentication with Oracle WebLogic Communication Services.

Note that the LDAP authentication provider and the Digest Authentication Identity Asserter provider can be configured with multiple LDAP servers to provide failover capabilities. If you want to use more than one LDAP server for failover, you will need to have connection information for each server when you configure Digest Authentication. See Section 5.7.4, "Steps for Configuring Digest Authentication".

Section 5.12, "Provisioning Resources in Oracle Internet Directory" describes how to provision the correct resources for Oracle Internet Directory.

Table 5–2 Digest Identity Asserter Checklist

Item	Description	Sample Value
Host	The host name of the LDAP server.	MyLDAPServer
Port	The port number of the LDAP server. Port 389 is used by default.	389
Principal	A Distinguished Name (DN) that Oracle WebLogic Communication Services can use to connect to the LDAP Server.	cn=ldapadminuser
Credential	A credential for the above principal name (generally a password).	ldapadminuserpassword
LDAP Connection Timeout	The configured timeout value for connections to the LDAP server (in seconds). For best performance, there should be no timeout value configured for the LDAP server. If a timeout value is specified for the LDAP server, you should configure the Digest Identity Asserter provider timeout to a value equal to or less than the LDAP server's timeout.	30 seconds
User From Name Filter	An LDAP search filter that Oracle WebLogic Communication Services will use to locate a given username. If you do not specify a value for this attribute, the server uses a default search filter based on the user schema.	(&(cn=%u)(objectclass=person))
User Base DN	The base Distinguished Name (DN) of the tree in the LDAP directory that contains users.	cn=users,dc=mycompany,dc=com
Credential Attribute Name	The credential attribute name used for Digest calculation. This corresponds to the attribute name used to store unencrypted passwords or pre-calculated hash values. See Section 5.7.4.1, "Configure the LDAP Server or RDBMS".	hashvalue
Digest Realm Name	The realm name to use for Digest authentication.	mycompany.com
Digest Algorithm	The algorithm that clients will use to create encrypted Digests. Oracle WebLogic Communication Services supports both MD5 and MD5-sess algorithms. MD5 is used by default.	MD5
Digest Timeout	The Digest authentication timeout setting. By default this value is set to 2 minutes.	2

5.7.4 Steps for Configuring Digest Authentication

Follow these steps to configure Digest authentication with Oracle WebLogic Communication Services:

1. Section 5.7.4.1, "Configure the LDAP Server or RDBMS".
2. Section 5.7.4.2, "Reconfigure the DefaultAuthenticator Provider".

Note: DefaultAuthenticator is set up as a required authentication provider by default. If the DefaultAuthentication provider, which works against the embedded LDAP store, is not used for authentication decisions, you must change the Control Flag to "SUFFICIENT".

3. Section 5.7.4.3, "Configure an Authenticator Provider".
4. Section 5.7.4.4, "Configure a New Digest Identity Asserter Provider".

The sections that follow describe each step in detail.

5.7.4.1 Configure the LDAP Server or RDBMS

The LDAP server or RDBMS used for Digest verification must store either unencrypted, clear text passwords, pre-calculated hash values, or passwords encrypted by a standard encryption algorithm (3DES_EDE/CBC/PKCS5Padding by default). The sections below provide general information about setting up your LDAP server or RDBMS to store the required information. Keep in mind that LDAP server uses different schemas and different administration tools, and you may need to refer to your LDAP server documentation for information about how to perform the steps below.

If you are using multiple LDAP servers to enable failover capabilities for the security providers, you must configure each LDAP server as described below.

5.7.4.1.1 Using Unencrypted Passwords If you are using an RDBMS, or if your LDAP server's schema allows storing unencrypted passwords in the user's password attribute, no additional configuration is needed. The Digest Identity Asserter provider looks for unencrypted passwords in the password field by default.

If the schema does not allow unencrypted passwords in the password attribute, you have two options:

- Store the unencrypted password in an existing, unused credential attribute in the LDAP directory.
- Create a new credential attribute to store the unencrypted password.

See your LDAP server documentation for more information about credential attributes available in the schema. Regardless of which method you use, record the exact attribute name used to store unencrypted passwords. You must enter the name of this attribute when configuring the LDAP Digest Identity Asserter provider.

5.7.4.1.2 Using Precalculated Hash Values If you want to use precalculated hash values, rather than unencrypted passwords, you can store the hash values in one of two places in your LDAP directory:

- In an existing, unused credential attribute.
- In a new credential attribute that you create for the hash value.

See your LDAP server documentation for more information using or creating new credential attributes.

For RDBMS stores, you can place the hash values in any column in your schema; you will define the SQL command used to obtain the hash values when configuring the RDBMS Identity Assertion Provider.

Oracle WebLogic Communication Services provides a simple utility (PreCalculatedHash) to generate a hash of the A1 value from a given username, realm name, and unencrypted password.

You can use also use 3rd-party utilities for generating the hash value, or create your own method using information from RFC 2617.

Note that you must also create the necessary infrastructure to update the stored hash value automatically when the user name, password, or realm name values change. Maintaining the password information in this manner is beyond the scope of this documentation.

5.7.4.1.3 Using Reverse-Encrypted Passwords Oracle WebLogic Communication Services provides a utility to help you compute the Encryption Key, Encryption Init Vector, and Encrypted Passwords values used when you configure the Digest Authorization Identity Asserter provider. The utility is named `JSafeEncryptionUtil` and is packaged in the `wlss.jar` file in the `WLSS_HOME/server/lib/wlss` directory.

To view usage instructions and syntax:

1. Add `wlss.jar` to your classpath. Here is the default location (but, it is user-customizable):

   ```
   export CLASSPATH=$CLASSPATH:WLSS_HOME=MW_HOME/wlcserver_
   10.3/server/lib/wlss/wlss.jar
   ```

2. Execute the utility without specifying options.

5.7.4.2 Reconfigure the DefaultAuthenticator Provider

In most production environments you will use a separate LDAP provider for storing password information, and therefore the DefaultAuthenticator, which works against the embedded LDAP store, must not be required for authentication. Follow the instructions in this section to change the provider's control flag to "sufficient".

> **Note:** DefaultAuthenticator is set up as a required authentication provider by default. If the DefaultAuthentication provider, which works against the embedded LDAP store, is not used for authentication decisions, you must change the Control Flag to "SUFFICIENT".

To reconfigure the DefaultAuthenticator provider:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. Select the *DefaultAuthenticator* provider.
6. In the **Configuration > Common** tab, change the Control Flag value to SUFFICIENT.
7. Click **Save** to save your changes.

5.7.4.3 Configure an Authenticator Provider

In addition to the Digest Identity Asserter providers, which only validate the client digest, you must configure an "authentication" provider, which checks for a user's existence and populates the user's group information. Follow the instructions in *Oracle Fusion Middleware Securing Oracle WebLogic Server* to create an LDAP authentication provider for your LDAP server. Use the information from Table 5–1, "Digest Authentication in Oracle WebLogic Communication Services" to configure the provider.

If you do not require user existence checking or group population, then, in addition to a Digest Identity Asserter provider, you can configure and use the special "no-op" authentication provider, packaged by the name "IdentityAssertionAuthenticator." This provider is helpful to avoid an extra round-trip connection to the LDAP server. Note that the provider performs no user validation and should be used when group information is not required for users.

To configure the "no-op" authorization provider:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. Click **New**.
6. Enter a name for the new provider, and select *IdentityAssertionAuthenticator* as the type.
7. Click **OK**.
8. Select the name of the new provider from the list of providers.
9. Set the Control Flag to SUFFICIENT in the **Configuration > Common** tab.
10. Click **Save** to save your changes.

5.7.4.4 Configure a New Digest Identity Asserter Provider

Follow these instructions in one of the sections below to create the Digest Identity Asserter provider and associate it with your LDAP server or RDBMS store:

- Section 5.7.4.4.1, "Configure an LDAP Digest Identity Asserter Provider"
- Section 5.7.4.4.2, "Configure an RDBMS Digest Identity Asserter Provider"

5.7.4.4.1 Configure an LDAP Digest Identity Asserter Provider

Follow these instructions to create a new LDAP Digest Identity Asserter Provider:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, myrealm).
4. Select the **Providers > Authentication** tab.
5. Click **New**.

6. Enter a name for the new provider, and select *LdapDigestIdentityAsserter* as the type.
7. Click **OK**.
8. Select the name of the new provider from the list of providers.
9. Select the **Configuration > Provider Specific** tab in the right pane.
10. On the configuration page, enter LDAP server and Digest authentication information into the fields as follows (use the information from Table 5–1):
    - **User From Name Filter**: Enter an LDAP search filter that Oracle WebLogic Communication Services will use to locate a given username. If you do not specify a value for this attribute, the server uses a default search filter based on the user schema.
    - **User Base DN**: Enter the base Distinguished Name (DN) of the tree in the LDAP directory that contains users (for example, cn=Users,dc=example,dc=com).
    - **Credential Attribute Name**: Enter the credential attribute in the LDAP directory that stores either the pre-calculated hash value or the unencrypted password (for example, authpassword;wlss). By default Oracle WebLogic Communication Services uses the password attribute of the user entry. If you use a pre-calculated has value instead of an unencrypted password, or if the unencrypted password is stored in a different attribute, you must specify the correct attribute name here.
    - **Group Attribute Name**: Enter the group attribute in the LDAP directory that stores a the set of group names to which the user belongs.
    - **Password Encryption Type**: Select the format in which the password is stored: `PLAINTEXT`, `PRECALCULATEDHASH`, or `REVERSIBLEENCRYPTED`.
    - **Encryption Algorithm**: If you have stored encrypted passwords, enter the encryption algorithm that the Digest identity assertion provider will use for reverse encryption.
    - **Encryption Key** and **Please type again to confirm**: If you have stored encrypted passwords, enter the base-64 encrypted key used as part of the reverse encryption algorithm.
    - **Encryption Init Vector** and **Please type again to confirm**: If you have stored encrypted passwords, enter the base-64 encrypted init vector string used as part of the reverse encryption algorithm.
    - **Digest Realm Name**: Enter the realm name to use for Digest authentication (for example, example.com).
    - **Digest Algorithm**: Select either MD5 or MD5-sess as the algorithm to use for encrypting Digests.
    - **Digest Timeout**: This value defines the nonce timeout value for the digest challenge. If the nonce timeout is reached before the client responds, the client is re-challenged with a new nonce. By default, the Digest Timeout is set to 120 seconds.
    - **Host**: Enter the host name of the LDAP server to use for Digest verification. If you are using multiple LDAP servers for failover capabilities, enter the *host name:port* value for each server separated by spaces. For example: `ldap1.mycompany.com:1050 ldap2.mycompany.com:1050`

See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information about configuring failover.

- **Port**: Enter the port number of the LDAP server.
- **SSL Enabled**: Select this option if you are using SSL to communicate unencrypted passwords between Oracle WebLogic Communication Services and the LDAP Server.
- **Principal**: Enter the name of a principal that Oracle WebLogic Communication Services uses to access the LDAP server (for example, orclApplicationCommonName=WLSSInstance1,cn=WLSS,cn=Products,cn=OracleContext,dc=example,dc=com).
- **Credential** and **Please type again to confirm**: Enter the credential for the above principal name (generally a password).
- **OIDSupportEnabled**: Select this checkbox if you are using Oracle Internet Directory as your LDAP provider. This checkbox is necessary when using a precalculated hash value because Oracle Internet Directory prefixes the hash value with {SASL/MD5} as described in RFC 2307. Other LDAP providers may omit the prefix.

11. Click **Save** to save your changes.
12. Select the **Performance** tab in the right pane.
13. On the Performance page, enter the caching and connection information into the fields as follows:
    - **LDAP Connection Pool Size**: Enter the number of connections to use for connecting to the LDAP Server. This value should be equal to or less than the total number of execute threads configured for Oracle WebLogic Communication Services. To view the current number of configured threads, right-click on the Oracle WebLogic Communication Services name in the left pane of the Administration Console and select View Execute Queues; the SIP Container uses the Thread Count value of the queue named sip.transport.Default. The default value of LDAP Connection Pool Size is 10.

      Note that stale connections (for example, LDAP connections that are timed out by a load balancer) are automatically removed from the connection pool.
    - **Cache Enabled**: Specifies whether a cache should be used with the associated LDAP server.
    - **Cache Size**: Specifies the size of the cache, in Kilobytes, used to store results from the LDAP server. By default the cache size is 32K.
    - **Cache TTL**: Specifies the time-to-live (TTL) value, in seconds, for the LDAP cache. By default the TTL value is 60 seconds.
    - **Results Time Limit**: Specifies the number of milliseconds to wait for LDAP results before timing out. Accept the default value of 0 to specify no time limit.
    - **Connect Timeout**: Specifies the number of milliseconds to wait for an LDAP connection to be established. If the time is exceeded, the connection times out. The default value of 0 specifies no timeout value.
    - **Parallel Connect Delay**: Specifies the number of seconds to delay before making concurrent connections to multiple, configured LDAP servers. If this value is set to 0, the provider connects to multiple servers in a serial fashion. The provider first tries to connect to the first configured LDAP server in the

Host list. If that connection attempt fails, the provider tries the next configured server, and so on.

If this value is set to a non-zero value, the provider waits the specified number of seconds before spawning a new thread for an additional connection attempt. For example, if the value is set to 2, the provider first tries to connect to the first configured LDAP server in the Host list. After 2 seconds, if the connection has not yet been established, the provider spawns a new thread and tries to connect to the second server configured in the Host list, and so on for each configured LDAP server.

- **Connection Retry Limit**: Specifies the number of times the provider tries to reestablish a connection to an LDAP server if the LDAP server throws an exception while creating a connection.

14. Click **Save** to save your changes.

5.7.4.4.2 Configure an RDBMS Digest Identity Asserter Provider Follow these instructions to create a new RDBMS Digest Identity Asserter Provider:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Click **Lock & Edit** to obtain a configuration lock.
3. In the left pane of the Console, select the **Security Realms** node.
4. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
5. Select the **Providers > Authentication** tab.
6. Click **New**.
7. Enter a name for the new provider, and select *DBMSDigestIdentityAsserter* as the type.
8. Click **OK**.
9. Select the name of the new provider from the list of providers.
10. Select the **Configuration > Provider Specific** tab in the right pane.
11. In the configuration tab, enter RDBMS server and Digest authentication information into the fields as follows:
    - **Data Source Name**: Enter the name of the JDBC DataSource used to access the password information.
    - **SQLGet Users Password**: Enter the SQL statement used to obtain the password or hash value from the database. The SQL statement must return a single record result set.
    - **SQLList Member Groups**: Enter a SQL statement to obtain the group information from a specified username. The username is supplied as a variable to the SQL statement, as in `SELECT G_NAME FROM groupmembers WHERE G_MEMBER = ?`.
    - **Password Encryption Type**: Select the format in which the password is stored: `PLAINTEXT`, `PRECALCULATEDHASH`, or `REVERSIBLEENCRYPTED`.
    - **Encryption Algorithm**: If you have stored encrypted passwords, enter the encryption algorithm that the Digest identity assertion provider will use for reverse encryption.

- **Encryption Key** and **Please type again to confirm**: If you have stored encrypted passwords, enter the base-64 encrypted key used as part of the reverse encryption algorithm.
- **Encryption Init Vector** and **Please type again to confirm**: If you have stored encrypted passwords, enter the base-64 encrypted init vector string used as part of the reverse encryption algorithm.
- **Digest Realm Name**: Enter the realm name to use for Digest authentication.
- **Digest Algorithm**: Select either MD5 or MD5-sess as the algorithm to use for encrypting Digests.
- **Digest Timeout**: This value defines the nonce timeout value for the digest challenge. If the nonce timeout is reached before the client responds, the client is re-challenged with a new nonce. By default, the Digest Timeout is set to 120 seconds.

12. Click **Save** to save your changes.

5.7.5 Sample Digest Authentication Configuration Using Embedded LDAP

You can use Oracle WebLogic Communication Services's embedded LDAP implementation for Digest authentication in a test or demo environment. Because you cannot change the schema of the embedded LDAP store, you must store password information in the existing "description" field.

To use the embedded LDAP store for Digest authentication, follow the instructions in the sections that follow.

5.7.5.1 Store User Password Information in the Description Field

To create new users with password information in the existing "description" field:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the Users and **Groups > Users** tab.
5. Click **New**.
6. Enter a name for the new user in the Name field.
7. Enter the Digest password information for the user in the Description field. The password information can be either the clear-text password, a pre-calculated hash value, or a reverse-encrypted password.
8. Enter an 8-character password in the Password and Confirm Password fields. You cannot proceed without adding a standard password entry.
9. Click **OK**.

5.7.5.2 Set the Embedded LDAP Password

Follow these instructions to set the password for the embedded LDAP store to a known password. You will use this password when configuring the Digest Identity Asserter provider as described in Section 5.7.4.4.1, "Configure an LDAP Digest Identity Asserter Provider":

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane, click the name of the domain you are configuring (for example, mydomain).
3. Select **Security > Embedded** LDAP in the right pane.
4. Enter the password you would like to use in the Credential and Confirm Credential fields.
5. Click **Save**.
6. Reboot the server.

5.7.5.3 Configure the Digest Identity Asserter Provider

Example 5–1 shows the security provider configuration in `config.xml` for a domain that uses LDAP implementation embedded in Oracle WebLogic Communication Services. Note that such a configuration is recommended only for testing or development purposes. Example 5–1 highlights values that you must define when configuring the provider using the instructions in Section 5.7.4.4.1, "Configure an LDAP Digest Identity Asserter Provider".

Example 5–1 Sample Security Provider Configuration with Embedded LDAP

```
<sec:authentication-provider
xmlns:ext="http://www.bea.com/ns/weblogic/90/security/extension"
xsi:type="ext:ldap-digest-identity-asserterType">
        <sec:name>myrealmLdapDigestIdentityAsserter</sec:name>
        <ext:user-base-dn>ou=people, ou=myrealm, dc=mydomain</ext:user-base-dn>
        <ext:credential-attribute-name>description</ext:credential-attribute-name>
        <ext:digest-realm-name>wlss.oracle.com</ext:digest-realm-name>
        <ext:host>myserver.mycompany.com</ext:host>
        <ext:port>7001</ext:port>
        <ext:principal>cn=Admin</ext:principal>
      </sec:authentication-provider>
```

5.8 Configuring Client-Cert Authentication

Client-Cert authentication uses a certificate or other custom tokens in order to authenticate a user. The token is "mapped" to a user present in the Oracle WebLogic Communication Services security realm in which the Servlet is deployed. SIP Servlets that want to use Client-Cert authentication must set the `auth-method` element to `CLIENT-CERT` in their `sip.xml` deployment descriptor.

The token used for Client-Cert authentication can be obtained in several different ways:

- **X509 Certificate from SSL**—In the most common case, an X509 certificate is derived from a client token during a two-way SSL handshake between the client and the server. The SIP Servlet can view the resulting certificate in the `javax.servlet.request.X509Certificate` request attribute. This method for performing Client-Cert authentication is the most common and is described in the SIP Servlet specification (JSR-116). Oracle WebLogic Communication Services provides two security providers that can be used to validate the X509 certificate; see Section 5.8.1, "Configuring SSL and X509 for Oracle WebLogic Communication Services".
- **WL-Proxy-Client-Cert Header**—Oracle WebLogic Communication Services provides an alternate method for supplying a Client-Cert token that does not

require a two-way SSL handshake between the client and server. Instead, the SSL handshake can be performed between a client and a proxy server or load balancer before reaching the destination Oracle WebLogic Communication Services. The proxy generates the resulting X509 certificate chain and encrypts it using base-64 encoding, and finally adds it to a special `WL-Proxy-Client-Cert` header in the SIP message. The server hosting the destination SIP Servlet then uses the `WL-Proxy-Client-Cert` header to obtain the certificate. The certificate is also made available by the container to Servlets via the `javax.servlet.request.X509Certificate` request attribute.

To use this alternate method of supplying client tokens, you must configure Oracle WebLogic Communication Services to enable use of the `WL-Proxy-Client-Cert` header; see Section 5.8.2, "Configuring Oracle WebLogic Communication Services to Use WL-Proxy-Client-Cert". You must also configure an X509 Identity Asserter provider as described in Section 5.8.1, "Configuring SSL and X509 for Oracle WebLogic Communication Services".

SIP Servlets can also use the `CLIENT-CERT auth-method` to implement perimeter authentication. Perimeter authentication uses custom token names and values, along with a custom security provider, to authenticate clients. See Section 5.8.3, "Supporting Perimeter Authentication with a Custom IA Provider" for a summary of steps required to implement perimeter authentication.

5.8.1 Configuring SSL and X509 for Oracle WebLogic Communication Services

Oracle WebLogic Communication Services includes two separate Identity Assertion providers that can be used with X509 certificates. The LDAP X509 Identity Asserter provider receives an X509 certificate, looks up the LDAP object for the user associated with that certificate in a separate LDAP store, ensures that the certificate in the LDAP object matches the presented certificate, and then retrieves the name of the user from the LDAP object. The Default Identity Asserter provider maps the user according to its configuration, but does not validate the certificate.

With either provider, Oracle WebLogic Communication Services uses two-way SSL to verify the digital certificate supplied by the client. You must ensure that a SIPS transport (SSL) has been configured in order to use Client-Cert authentication. See Managing Oracle WebLogic Communication Services Network Resources in *Configuring Network Resources* if you have not yet configured a secure transport.

See Section 5.8.1.1, "Configuring the Default Identity Asserter" to configure the Default Identity Asserter provider. In most production installations you will have a separate LDAP store and will need to configure the LDAP X509 Identity Asserter provider to use client-cert authentication; see Section 5.8.1.2, "Configuring the LDAP X509 Identity Asserter".

5.8.1.1 Configuring the Default Identity Asserter

The Default Identity Asserter can be configured to verify an X509 certificate passed to it by a client over a secure (SSL) connection. The Default Identity Asserter requires a separate user name mapper to map the associated client "certificate" to a user configured in the default security realm. You can use the default user name mapper installed with Oracle WebLogic Communication Services, or you can create a custom user name mapper class as described in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Follow these instructions to configure the Default Identity Asserter:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.

2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. In the right pane of the Console, select *DefaultIdentityAsserter* from the table of configured providers.
6. On the **Configuration > Common** page, select X.509 in the Available column of the Active Types table and use the arrow to move it to the Chosen column.
7. Click **Save** to apply the change.
8. You can use either a custom Java class to map names in the X509 certificate to usernames in the built-in LDAP store, or you can use the default user name mapper. To specify a custom Java class to perform user name mapping:
   a. Select the **Configuration > Provider Specific** tab.
   b. Enter the name of the custom class in the User Name Mapper Class Name field.
   c. Click **Save**.

   To use the default user name mapper:
   a. Select the **Configuration > Provider Specific** tab.
   b. Select *Use Default User Name Mapper*.
   c. In the Default User Name Mapper Attribute Type list, select either CN (for Common Name) or E (for Email address) depending on the user name attribute you have stored in the security realm.
   d. In the Default User Name Mapper Attribute Delimiter field, accept the default delimiter of "@". This delimiter is used with the E (Email address) attribute type to extract the email portion from the client token. For example, a token of "joe@mycompany.com" would be mapped to a username "joe" configured in the default security realm.
   e. Click **Save**.

5.8.1.2 Configuring the LDAP X509 Identity Asserter

Follow these steps to create and configure the X509 Authentication Provider.

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. Click **New**.
6. Enter a name for the new provider, and select *LDAPX509IdentityAsserter* as the type.
7. Click *OK*.
8. In the list of providers, select the name of the provider you just created.

9. In the **Configuration > Provider Specific** tab, enter LDAP server information into the fields as follows:

   - **User Field Attributes**: Enter an LDAP search filter that Oracle WebLogic Communication Services will use to locate a given username. The filter is applied to LDAP objects beneath the base DN defined in the **Certificate Mapping** attribute described below.
   - **User Name Attribute**: Enter the LDAP attribute that stores the user's name.
   - **Certificate Attribute**: Enter the LDAP attribute that stores the certificate for the user name.
   - **Certificate Mapping**: Specify how a query string to construct the base LDAP DN used to locate the LDAP object for the user.
   - **Host**: Enter the host name of the LDAP server to verify the incoming certificate. If you are using multiple LDAP servers for failover capabilities, enter the *host name:port* value for each server separated by spaces. For example: `ldap1.mycompany.com:1050 ldap2.mycompany.com:1050`

     See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information about configuring failover.
   - **Port**: Enter the port number of the LDAP server.
   - **SSL Enabled**: Select this option if you are using SSL to communicate unencrypted passwords between Oracle WebLogic Communication Services and the LDAP Server.
   - **Principal**: Enter the name of a principal that Oracle WebLogic Communication Services uses to access the LDAP server.
   - **Credential**: Enter the credential for the above principal name (generally a password).
   - **Confirm Credential**: Re-enter the principal's credential.
   - **Cache Enabled**: Specifies whether a cache should be used with the associated LDAP server.
   - **Cache Size**: Specifies the size of the cache, in Kilobytes, used to store results from the LDAP server. By default the cache size is 32K.
   - **Cache TTL**: Specifies the time-to-live (TTL) value, in seconds, for the LDAP cache. By default the TTL value is 60 seconds.
   - **Follow Referrals**: Select this to specify that a search for a user or group within the LDAP X509 Identity Assertion provider should follow referrals to other LDAP servers or branches within the LDAP directory.
   - **Bind Anonymously On Referrals**: By default, the LDAP X509 Identity Assertion provider uses the same DN and password used to connect to the LDAP server when following referrals during a search. If you want to connect as an anonymous user, check this box.
   - **Results Time Limit**: Specifies the number of milliseconds to wait for LDAP results before timing out. Accept the default value of 0 to specify no time limit.
   - **Connect Timeout**: Specifies the number of milliseconds to wait for an LDAP connection to be established. If the time is exceeded, the connection times out. The default value of 0 specifies no timeout value.
   - **Parallel Connect Delay**: Specifies the number of seconds to delay before making concurrent connections to multiple, configured LDAP servers. If this

value is set to 0, the provider connects to multiple servers in a serial fashion. The provider first tries to connect to the first configured LDAP server in the Host list. If that connection attempt fails, the provider tries the next configured server, and so on.

If this value is set to a non-zero value, the provider waits the specified number of seconds before spawning a new thread for an additional connection attempt. For example, if the value is set to 2, the provider first tries to connect to the first configured LDAP server in the Host list. After 2 seconds, if the connection has not yet been established, the provider spawns a new thread and tries to connect to the second server configured in the Host list, and so on for each configured LDAP server.

- **Connection Retry Limit**: Specifies the number of times the provider tries to reestablish a connection to an LDAP server if the LDAP server throws an exception while creating a connection.

10. Click **Save** to save your changes.
11. Reboot the server to realize the changed security configuration.

5.8.2 Configuring Oracle WebLogic Communication Services to Use WL-Proxy-Client-Cert

In order for Oracle WebLogic Communication Services to use the `WL-Proxy-Client-Cert` header, a proxy server or load balancer must first transmit the X509 certificate for a client request, encrypt it using base-64 encoding, and then add the resulting token `WL-Proxy-Client-Cert` header in the SIP message. If your system is configured in this way, you can enable the local Oracle WebLogic Communication Services instance (or individual SIP Servlet instances) to examine the `WL-Proxy-Client-Cert` header for client tokens.

To configure the server instance to use the `WL-Proxy-Client-Cert` header:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane, select the Environment > Servers node.
3. Select the name of a configured engine tier server.
4. Select the Configuration > General tab in the right pane.
5. Select Client Cert Proxy Enabled.
6. Click Save to save your changes.
7. Follow the instructions under Section 5.8.1, "Configuring SSL and X509 for Oracle WebLogic Communication Services" to configure either the default identity asserter or the LDAP Identity Asserter provider to manage X509 certificates.
8. Reboot the server to realize the changed configuration.

To enable the `WL-Proxy-Client-Cert` header for an individual Web Application, set the `com.bea.wcp.clientCertProxyEnabled` context parameter to true in the application's `sip.xml` deployment descriptor.

5.8.3 Supporting Perimeter Authentication with a Custom IA Provider

With perimeter authentication, a system outside of WebLogic Server establishes trust via tokens. The system is generally comprised of an authentication agent that creates an artifact or token that must be presented to determine information about the

authenticated user at a later time. The actual format of the token varies from vendor to vendor (for example, SAML or SPNEGO).

Oracle WebLogic Communication Services supports perimeter authentication through the use of an Identity Assertion provider designed to recognize one or more token formats. When the authentication type of a SIP Servlet is set to `CLIENT-CERT`, the SIP container in Oracle WebLogic Communication Services performs identity assertion on values from the request headers. If the header name matches the active token type for a configured provider, the value is passed to the provider for identity assertion.

The provider can then use a user name mapper to resolve the certificate to a user available in the security realm. The user corresponding to the Subject's Distinguished Name (SubjectDN) attribute in the client's digital certificate must be defined in the server's security realm; otherwise the client will not be allowed to access a protected WebLogic resource.

If you want to use custom tokens to pass client certificates for perimeter authentication, you must create and configure a custom Identity Assertion provider in place of the LDAP X509 or Default Identity Asserter providers described above. See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for information about creating providers for handling tokens passed with perimeter authentication.

5.9 Configuring SIP Servlet Identity Assertion Mechanisms

A SIP Servlet can be configured to use one of the following identity assertion mechanisms:

- `P-Asserted-Identity`: With this mechanism, identity must be asserted using the `P-Asserted-Identity` header in a SIP message that originates from a trusted domain. This identity assertion mechanism is described in RFC 3325.
- `Identity`: With this mechanism, identity must be asserted using the `Identity` and `Identity-Info` headers in SIP messages, which can originate from other domains. This identity assertion mechanism is described in RFC 4474.

The selected identity assertion mechanism is defined in the `identity-assertion` element of the `sip.xml` deployment descriptor. The `identity-assertion-support` element determines whether the identity assertion mechanism is required for the Servlet, or whether alternate authentication mechanisms can be used with SIP messages that do not contain the required headers. See the SIP Servlet Specification v1.1 for more information on configuring identity assertion for a Servlet.

Oracle WebLogic Communication Services supports identity assertion mechanisms using security providers. The sections that follow describe how Oracle WebLogic Communication Services handles messages with each identity assertion mechanism, and how to configure the required security providers.

Note: Oracle WebLogic Communication Services version provides backward compatibility for applications that conform to the SIP Servlet v1.0 specification.

5.9.1 Understanding Trusted Host Forwarding with P-Asserted-Identity

The `P-Asserted-Identity` header is honored only within a trusted domain. In a Oracle WebLogic Communication Services system, trusted domains are purely configuration-based. To enable use of the header, you must configure one of two available P-Asserted Identity Assertion providers as described in Section 5.9.3,

"Configuring a P-Asserted-Identity Assertion Provider". The `P-Asserted-Identity` assertion providers expose the trusted domain configuration for `P-Asserted-Identity` headers. If you do not configure a provider, the header considers no IP addresses as being "trusted."

When Oracle WebLogic Communication Services receives a message having the `P-Asserted-Identity` header from a trusted host configured with the provider, it logs in the user specified in the header to determine group membership and other privileges. The value contained in the `P-Asserted-Identity` header must be a SIP address (for example, `sipuser@oracle.com`). By default, Oracle WebLogic Communication Services removes the domain portion of the address (`@oracle.com`) and uses the remainder as the user name. If you must support overlapping usernames from different names (for example, `sipuser@oracle.com` and `sipuser@cea.com`), you can create and use a custom user-name mapper to process the header contents into a unique username (for example, `sipsuser_b` and `sipuser_c`). Using a custom user name mapper also enables you to support WebLogic user names that contain an "@" character, such as `@oracle.com`.

The presence of a `P-Asserted-Identity` header combined with the `Privacy` header also determines the way in which Oracle WebLogic Communication Services proxies incoming requests. The value of the `identity-assertion-support` element in `sip.xml` is also considered. Figure 5–3 describes how incoming SIP requests are managed in relation to the `P-Asserted-Identity` header.

Figure 5–3 Managing Inbound Requests Having P-Asserted-Identity and Privacy Headers

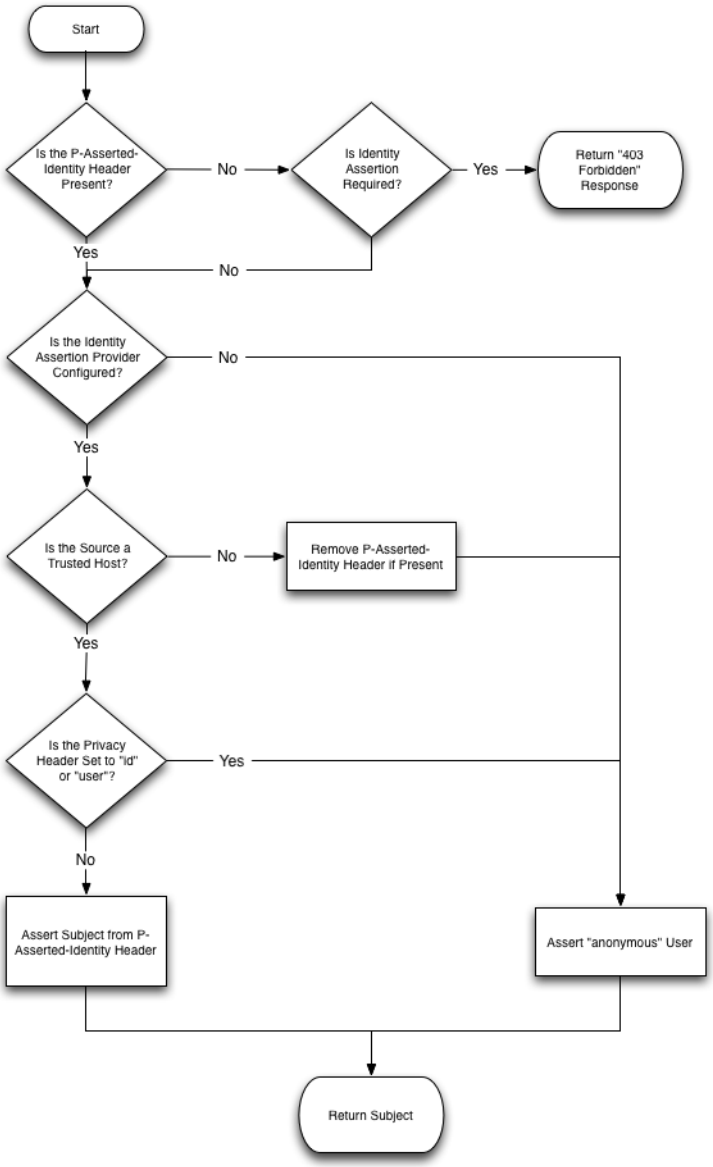
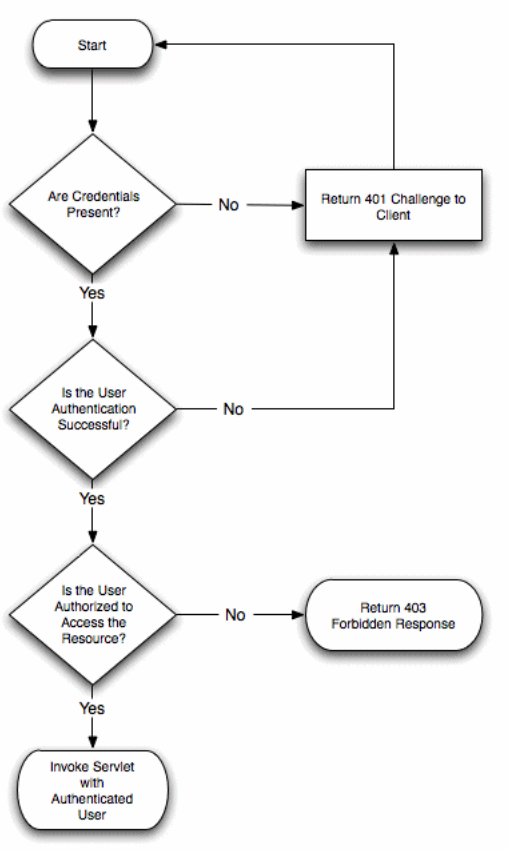


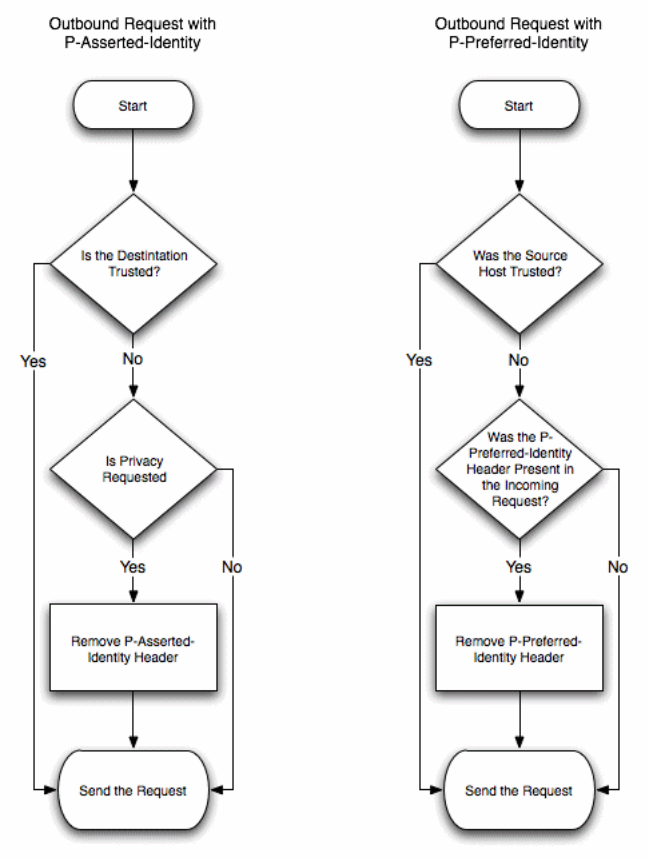
Figure 5–4 describes the standard security check procedure that Oracle WebLogic Communication Services uses when an asserted user name is not authorized to access a requested resource. The standard security check is performed according to the `auth-method` defined in the `login-config` element of the `sip.xml` descriptor for the current application.

Figure 5–4 Standard Security Check Procedure



The presence of a `P-Asserted-Identity` header or a `P-Preferred-Identity` header also affects the processing of outbound SIP requests. Figure 5–5 describes the behavior.

Figure 5–5 Managing Outbound Requests Having P-Asserted-Identity or P-Preferred Identity



5.9.2 Overview of Strict and Non-Strict P-Asserted-Identity Asserter Providers

If the contents of a P-Asserted-Identity header are invalid, or if the header is received from a non-trusted host, then the security provider returns an "anonymous" user to the SIP Servlet container. If you configured the **PAsserted Identity Strict Asserter** provider, an exception is also thrown so that you can audit the substitution of the anonymous user. (If you configured the basic **PAsserted Identity Asserter** provider, no exception is thrown.)

With either provider, if identity assertion fails and the requested resource is protected (the request matches a `security-constraint` defined in `sip.xml`), the SIP container uses the `auth-method` defined in the `sip.xml` deployment descriptor to challenge the end user. For example, digest authentication may be used if the Servlet specifies the digest authentication method.

If the requested resource is not protected, the anonymous user is simply passed to the SIP Servlet without authorization. Because the 3GPP TS 24.229 specification recommends forced authorization even when a resource is unrestricted (and privacy is not requested), you should use declarative security to protect all of a SIP Servlet's resources to remain compliant with the specification.

If authorization of the anonymous user fails, Oracle WebLogic Communication Services then forces authentication by challenging the user.

5.9.3 Configuring a P-Asserted-Identity Assertion Provider

Follow these steps to configure a security provider used to support the `P-Asserted-Identity` header. Note that one of two providers can be selected, as described in Section 5.9.2, "Overview of Strict and Non-Strict P-Asserted-Identity Asserter Providers".

In addition to configuring one of the above providers, configure a secondary, "fallback" login method (for example, using DIGEST or CLIENT-CERT authentication).

To configure a `P-Asserted-Identity` provider:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Click **New**.
5. Enter a name for the new provider, and select one of the following options for the Type:
   - PAssertedIdentityAsserter—Select this option to configure a provider that does not throw an exception when the `P-Asserted-Identity` header is invalid or is received from a non-trusted host and an anonymous user is substituted.
   - PAssertedIdentityStrictAsserter—Select this option to configure a provider that throws an exception when the `P-Asserted-Identity` header is invalid or is received from a non-trusted host and an anonymous user is substituted.

   See Section 5.9.2, "Overview of Strict and Non-Strict P-Asserted-Identity Asserter Providers" for more information.
6. Click **OK**.
7. Select the name of the provider you just created.
8. Select the **Configuration > Provider Specific** tab.
9. Fill in the fields of the configuration tab as follows:
   - **Trusted Hosts**: Enter one or more host names that the provider will treat as trusted hosts. You can enter a list of IP addresses or DNS names, and wildcards are supported.

     Note: The provider *does not use* trusted hosts configured in the `sipserver.xml` file.

   - **User Name Mapper Class Name**: Enter the name of a custom Java class used to map user names in the `P-Asserted-Identity` header to user names in the default security realm. A custom user name mapper is generally used if user names are received from two or more different domains. In this case additional logic may be required to map usernames received from each domain. A custom user name mapper class is required if you want to map usernames in the `P-Asserted-Identity` header to WebLogic usernames. See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information.

Alternately, leave this field blank to use the default user name mapper. The default mapper simply discards the domain name and takes the resulting user name without applying any additional logic.

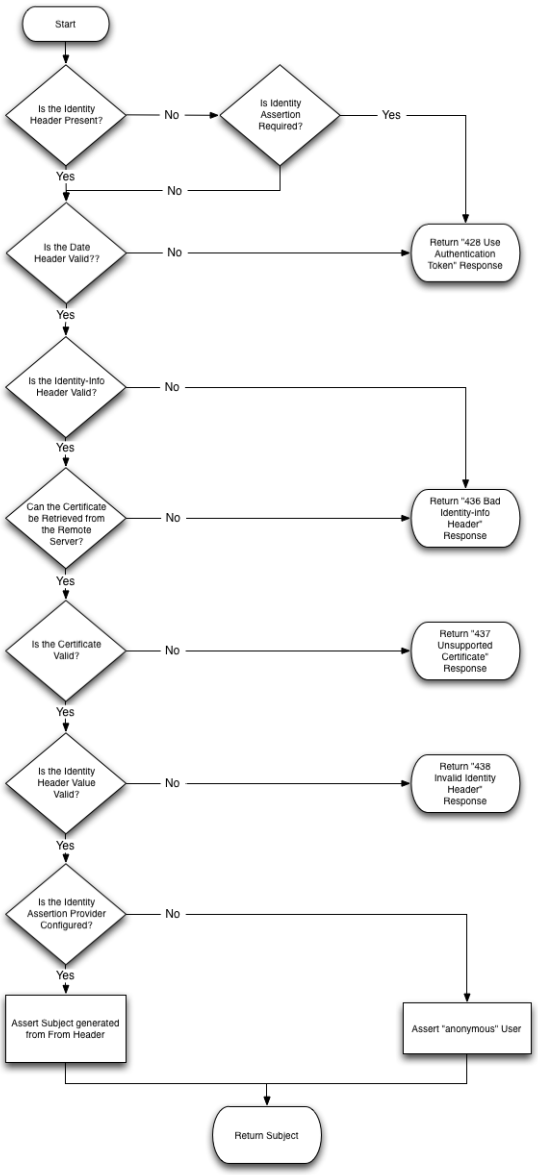
10. Click **Save**.

5.9.4 Understanding Identity Assertion with the Identity and Identity-Info Headers

Oracle WebLogic Communication Services can also perform identity assertion using the `Identity` and `Identity-Info` headers, as described in RFC 4474. As with the `p-asserted-identity` assertion mechanism, `Identity` header assertion requires that you first configure the appropriate security provider (the `IdentityHeaderAsserter` provider) in Oracle WebLogic Communication Services.

When asserting the identity of inbound requests having the `Identity` and `Identity-Info` headers, Oracle WebLogic Communication Services considers the values of the `identity-assertion` and `identity-assertion-support` elements in `sip.xml` as well as the presence of a configured security provider. Figure 5–6 describes how incoming messages are processed using this assertion mechanism.

Figure 5–6 Managing Inbound Requests Having Identity and Identity-Info Headers



5.9.5 Configuring the Identity Header Assertion Provider

Follow these steps to configure the security provider used to support the `Identity` header:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Click the **Providers > Authentication** tab in the right pane.
5. Click **New**.

6. Enter a name for the new provider, and select `IdentityHeaderAsserter` for the Type.
7. Click **OK**.
8. Select the name of the provider you just created.
9. Select the **Provider Specific** tab.
10. Fill in the fields of the configuration tab as follows:
    - **Date Period**: Enter the valid period for Date header, in seconds.
    - **Https Channel Name**: Enter the name of an HTTPS channel the provider should use to initialize an HTTPS client. An HTTPS channel is required (and must be configured separately) if a remote certificate must be retrieved via HTTPS.
    - **User Name Mapper Class Name** (optional): Enter the name of a custom Java class used to map user names in the `Identity` header to user names in the default security realm. A custom user name mapper class is required if you want to map usernames in the `Identity` header to WebLogic usernames. See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information.
11. Click **Save**.

5.10 Configuring 3GPP HTTP Identity Assertion Providers

In order to function as an Application Server in an IMS network, Oracle WebLogic Communication Services supports handling the `X-3GPP-Asserted-Identity` header as specified in 3GPP TS 33.222 Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (`http://www.3gpp.org/ftp/Specs/html-info/33222.htm`). Oracle WebLogic Communication Services provides this support via a configured security provider, `X3gppAssertedIdentityAsserter` or `X3gppAssertedIdentityStrictAsserter`. The providers use the same authentication process, but the "strict" assertion provider also throws an exception when the header is received from a non-trusted host (which enables you to audit asserted identity requests from non-trusted hosts).

The `X-3GPP-Asserted-Identity` header functions for HTTP requests in the same manner that the `P-Asserted-Identity` header functions for SIP requests. When the container receives an incoming HTTP requesting having a `X-3GPP-Asserted-Identity` header, it first verifies that the request was received from a trusted host. If the host was trusted, the container asserts the user's identity using the information in the header, authenticates the user, and logs the user in if that user is authorized to access the requested resource. (If a request comes from a non-trusted host, the container simply ignores the header.)

The `X-3GPP-Asserted-Identity` header may contain multiple names in a list (for example, user1@oracle.com, user2@oracle.com). When configured with the default user name mapper class, the Oracle WebLogic Communication Services providers remove the domain portion of the addresses (`@oracle.com`) and use the remainder as the user name. The default user name mapper always chooses the first username in the list and uses it for asserting the identity. This behavior can be changed by creating and configuring a custom user name mapper class. For example, if you must support overlapping usernames from different names (for example, `sipuser@oracle.com` and `sipuser@cea.com`), a custom user-name mapper might process the header contents into a unique username (for example, `sipsuser_b` and `sipuser_c`). Using

a custom user name mapper also enables you to support WebLogic user names that contain an "@" character, such as `@oracle.com`.

In order for SIP Servlets to support authentication with the `X-3GPP-Asserted-Identity` header, the `auth-method` element must be set to `CLIENT-CERT` in the `web.xml` deployment descriptor. See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information.

5.10.1 Configuring a X-3GPP-Asserted-Identity Provider

Follow these steps to configure a security provider used to support the `X-3GPP-Asserted-Identity` header in HTTP requests. Note that one of two providers can be selected, as described in the Section 5.10, "Configuring 3GPP HTTP Identity Assertion Providers":

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. Click **New**.
6. Enter a name for the new provider, and select one of the following options for the Type field:
   - X3gppAssertedIdentityAsserter—Select this option to configure a provider that does not throw an exception when the header is invalid or is received from a non-trusted host.
   - X3gppAssertedIdentityStrictAsserter—Select this option to configure a provider that throws an exception when the header is received from a non-trusted host and is therefore ignored.

   See Section 5.10, "Configuring 3GPP HTTP Identity Assertion Providers" for more information.
7. Click **OK**.
8. Select the name of the new provider you just created.
9. In the Active Types Chooser list, select the *X-3GPP-Asserted-Identity* type and use the arrow to move it to the Chosen column.
10. Click **Save**.
11. Select the **Configuration > Provider Specific** tab.
12. Fill in the fields of the configuration page as follows:
    - **Trusted Hosts**: Enter one or more host names that the provider will treat as trusted hosts. Note that the provider *does not use* trusted hosts configured in the `sipserver.xml` file (see sip-security *in the Configuration Reference Manual*.) You can enter a list of IP addresses or DNS names, and wildcards are supported.
    - **User Name Mapper Class Name**: Enter the name of a custom Java class used to map user names in the `X-3GPP-Asserted-Identity` header to user names in the default security realm. A custom user name mapper is generally used if user names are received from two or more different domains. In this

case additional logic may be required to map user names received from each domain. A custom user name mapper class is required if you want to map usernames to WebLogic usernames, or if you want to logically process multiple usernames specified in the `X-3GPP-Asserted-Identity` header (rather than using only the first username). See *Oracle Fusion Middleware Securing Oracle WebLogic Server* for more information.

Alternately, leave this field blank to use the default user name mapper. The default mapper simply discards the domain name and takes the first resulting user name to assert the identity. For example, the default user name mapper takes the following header:

```
X-3GPP-Asserted-Identity: "user1@oracle.com", "user2@oracle.com"
```

and asserts the identity "user1."

13. Click **Save**.

5.11 Configuring Basic Authentication for HTTP Servlets

Although Basic authentication is deprecated for use with SIP Servlets, you may choose to use this authentication mechanism with HTTP Servlets. Basic authentication is supported via the LDAPAuthenticator provider. Follow these steps to configure the provider with Oracle Internet Directory:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. In the left pane of the Console, select the **Security Realms** node.
3. Select the name of your security realm in the right pane of the Console. (for example, *myrealm*).
4. Select the **Providers > Authentication** tab.
5. Click **New**.
6. Enter a name for the new provider, and select *LDAPAuthenticator* for the Type field.
7. Click **OK**.
8. Select the name of the new provider you just created.
9. Select the **Configuration > Provider Specific** tab.
10. Fill in the fields of the configuration page as follows:
    - **Principal**: Enter the application instance that was configured for Oracle WebLogic Communication Services in Oracle Internet Directory (for example, orclApplicationCommonName=WLSSInstance1,cn=WLSS,cn=Products,cn=OracleContext,dc=example,dc=com). Note that you must provision this instance manually after installing Oracle Internet Directory. See Section 5.12, "Provisioning Resources in Oracle Internet Directory" for instructions.
    - **Credential**: Enter the password of the Principal that was configured in Oracle Internet Directory.
    - Group Base DN: Enter the DN of the Groups object in Oracle Internet Directory (for example, cn=groups,dc=example,dc=com).
    - User Base DN: Enter the DN of the Users object in Oracle Internet Directory (for example, cn=Users,dc=example,dc=com).
11. Click **Save**.

5.12 Provisioning Resources in Oracle Internet Directory

The following sections provide an overview of how to provision Oracle WebLogic Communication Services resources when using Oracle Internet Directory as your LDAP provider. These instructions are necessary when using Digest Authentication with a precalculated hash value, or when configuring Basic authentication for HTTP Servlets.

See the *Oracle Fusion Middleware Administrator's Guide* for Oracle Internet Directory for more details about these procedures.

5.12.1 Configuring Oracle Internet Directory

You must configure the following mappings for the OID LDAP backend:

- JAAS Usernames to LDAP User Entries--JAAS (Java Authentication and Authorization Service) user names are mapped to LDAP Users based on value of the orclcommonnicknameattribute under the node cn=Common,cn=Products,cn=OracleContext. For example, setting this attribute to uid implies that users authenticating against OID must provide their corresponding LDAP uid as their username during authentication. The rest of the configuration described in this chapter assumes that the orclcommonnicknameattribute is set to uid (default value).
- JAAS Realms to LDAP Subscribers--JAAS realms are mapped to LDAP Realm entries based on the value given to orclsubscribernicknameattribute under the root cn=Common,cn=Products,cn=OracleContext node for an OID deployment. For example, setting the value of orclsubscribernicknameattribute to o (the letter "o") for an OID deployment implies that users authenticating against OID must belong to the JAAS realm identified by the value of the o attribute. Set the value of orclsubscribernicknameattribute to o.
- JAAS Roles to LDAP Groups--Group membership determines the JAAS roles for a specific user. Mapping LDAP groups to JAAS roles is based on the value given to orclcommonnamingattribute under the node cn=Common,cn=Products,cn=OracleContext for each of the provisioned LDAP Realms. For example, if a user belongs to an LDAP group with the distinguished name of cn=Location Service, cn=groups, dc=example, dc=com and the orclcommonnamingattribute is set to cn, then that JAAS user is populated with the "Location Service" JAAS role. Set the value of orclcommonnamingattribute to cn

5.12.2 Configuring Static Verifiers

After configuring Oracle Internet Directory as described above, you must create a new product entry for Oracle WebLogic Communication Services (OWLCS), install the static verifier, create entries for each instance of OWLCS, and grant verifier privileges to each new instance created. You must perform these steps before provisioning users in OID. If users already exist in OID, after creating and configuring the static verifier, users must reset their passwords before they can login successfully.

5.12.2.1 Add Oracle WebLogic Communication Services

To add the Oracle WebLogic Communication Services product to Oracle Internet Directory:

1. Start the oidadmin tool in $ORACLE_HOME/bin/ and connect to the installed Oracle Internet Directory server. Login using "orcladmin" account and the password you chose during your installation of Oracle Internet Directory.

2. Browse the Entry Management tree to find:

   cn=Products,cn=OracleContext,dc=example,dc=com. The exact domain part (dc=example,dc=com) will depend on the domain that you created when you installed Oracle Internet Directory

3. A convenient way to create a new entry for OWLCS is to clone an existing product entry. Select the first entry under Products (this is generally the "Calendar" entry) and right-click on the Calendar entry and choose "Create Like". In the resulting dialog:

   Replace "Calendar" entry in dn with WLCS.

   Enter WLCS for cn

4. Click **OK**.

5. Select the Products entry, right-click and choose "Refresh SubTree Entries" and make sure that a new product with name WLCS shows up under Products.

5.12.2.2 Install the Static Verifier

Use the ldapadd command-line tool to install the static verifier, as follows:

1. Set the environmental variable ORACLE_HOME to point to the ORACLE_HOME of your OID installation

2. Create an ldif file containing the following lines (again replace the domain parts with your domain):

   dn:
   cn=WLCSVerifierProfileEntry,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com

   objectclass:top

   objectclass:orclpwdverifierprofile

   cn:WLCSVerifierProfileEntry

   orclappid:wlcs

   orclpwdverifierparams;authpassword: crypto:SASL/MD5 $ realm:example.com $ usernameattribute:uid

3. cd $ORACLE_HOME

4. Run the command ./bin/ldapadd -D cn=orcladmin -w <password of orcladmin user> -f <yourfile>.ldif

5. In oidadmin, refresh the WLCS product entry (by right-clicking on entry and choosing "Refresh SubTree Entries"). The WLCSVerifierProfileEntry should appear

5.12.3 Add a New Oracle WebLogic Communication Services

To add a new Oracle WebLogic Communication Services Instance:

1. Select the WLCS product entry you created, right-click and choose **Create**.

2. In the Distinguished Name (dn) field enter

   ```
   orclApplicationCommonName=WLCSInstance1,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com" (replacing the domain part with your domain)
   ```

3. Under Object Classes, click **Add**.

4. Select the "orclApplicationEntity". Click **Select**.
5. Click on the Optional Properties and populate the values of the following attributes:

   userpassword (not authpassword) - enter any password of your choice

   orclappfullname - enter "Oracle Weblogic Communication Services"

   description - enter "Entry for Oracle Weblogic Communication Services Instance"
6. Click OK.
7. Refresh the WLCS product by right-clicking the entry and choosing "Refresh SubTree Entries" and make sure that you see the new entry you just created

5.12.4 Grant Verifier Privileges to the Oracle WebLogic Communication Services Instance

To grant Verifier Privileges to the Oracle WebLogic Communication Services Instance:

1. Navigate to: cn=verifierServices,cn=Groups,cn=OracleContext,dc=example,dc=com entry (replacing the domain part with your domain).
2. Click cn=verifierServices
3. In the right pane, scroll down to the uniquemember attribute. You might see an entry or two for the value of the attribute. Add "orclApplicationCommonName=WLCSInstance1,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com" (replacing the domain part with your domain)to the existing value of uniquemember attribute.
4. Click **Apply**.
5. Repeat above two steps for each instance of OWLCS

   For each instance of OWLCS that needs to communicate with OID, you need to repeat the above two steps (Adding a new OWLCS instance and granting verifier privileges to the instance)

5.13 Provisioning Users

To provision users, you have to first create a user, set required attributes for the user, create a group, and assign the new user to be a member of the group by doing the following:

5.13.1 Create a New User

See the OID manual for administrating users via oiddas.

Alternatively, if you want to quickly create one test user you can use the oidadmin and clone the orcladmin user as follows:# Navigate to cn=Users,dc=example,dc=com (replacing the domain part with your domain)

1. Right-click on cn=orcladmin and choose **Create Like**.
2. In the resulting dialog, do the following:

   Change orcladmin in the Distinguised Name attribute value to "test.user1"

   Enter test.user1 for cn

   Enter test.user1 for sn

3. Click on the "Optional Properties" tab

   Enter test.user1 for givenName

   Enter test.user1@example.com for mail (since this email address will be the user's SIP address also, make sure you enter it in the form username@your-domain)

   Change description to "Test User for OWLCS"

   Change orclSAMAccountName to test.user1

   Enter test.user1 for uid

   Enter a password for this test user of your choice

   Enter true for orclIsVisible property (this is important)

4. Click OK.

5.13.2 Create a Group

If you want to run the ProxyRegistrar then all users must be members of the 'Location Service' group. That group has to be created, as described below.

The easiest way to add groups and add members to groups is via the OIDDAS web interface. Consult the OID documentation on how to create groups. Alternatively, you can create a new group by cloning an existing group using the oidadmin tool by doing the following:

1. Browse to Entry Management tree until you get to cn=Groups,dc=example,dc=com (replacing the domain part with your domain)
2. Right-click on an existing Group (such as "OCS_Portal_Users") and choose **Create Like**.
3. Set the Distinguished Name (dn) to cn=Location Service,cn=Groups,dc=example,dc=com (change OCS_Portal_Users to Location Service)
4. Enter Location Service for cn.
5. Click on the **Optional Properties** tab.
6. For the description, enter "Location Service Role for OWLCS".
7. Enter Location Service for displayName.
8. Click **OK**.
9. Click on cn=Groups and choose "Refresh SubTree Entries" and make sure you see the new group.

5.13.3 Assign Group Memberships to Users

As already mentioned, if you want to run the ProxyRegistrar then all OWLCS users must be members of the 'Location Service' group created above. To add a user to the new Location Service group, do the following:

1. Click on the new Location Service group you created. You will find the group in the Entry Management tree at cn=Location Service,cn=Groups,dc=example,dc=com (replace domain part with your domain)
2. In the pane on your right, scroll down to uniquemember attribute and add cn=test.user1,cn=Users,dc=example,dc=com to any existing entries (replace domain part with your domain).

3. Click **Apply**.

5.13.4 Set JAAS Realm for Users

Next you must set the JAAS Realm for users:

1. Navigate to dc=example,dc=com under Entry Management (replace domain with your domain
2. Click the dc=example entry.
3. Click the **Advanced** radio button.
4. Select o from the Attribute drop down.
5. Click on the **Apply** button.
6. Enter example.com for the value of the o attribute (replace with your realm).
7. Click **Apply**.

5.14 Configuring OWLCS Server Instance

Add an LDAPIdentityAssertionProvider with OID support:

1. In the WLS Admin Console go to your Security Realm and click the **Providers** tab.
2. Delete the DigestIdentityAsserter if you see it in the list of providers (this is created by default for out-of-the-box installation of OWLCS) and restart OWLCS server.
3. After server restarts, in the WLS Admin Console, click on the Providers tab.
4. Add a new LDAP Digest Identity Assertion Provider (by clicking the **New** button and selecting *LDAPDigestAssertionProvider* from the Type drop-down). Enter *LDAPDigestAssertionProvider* for name. Click **OK**.
5. Click on the **Provider Specific** tab.
6. Set the UserBaseDN to "cn=Users,dc=example,dc=com" (replace the domain part with your domain).
7. Set the CredentialAttributeName to "authpassword;wlcs".
8. Set the PaswordEncryptionType to PRECALCULATEDHASH.
9. Set the DigestRealmName to the "example.com" (this should match the realm value in your ldif file used for installing static verifier).
10. Set Host and Port to those of the OID server.
11. Set the Principal to "orclApplicationCommonName=WLCSInstance1,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com" (replace domain part with your domain)
12. Set the Credential to what you configured the userPassword of the instance above. Confirm credential.
13. Check the "OIDSupportEnabled" checkbox and click **Save**.
14. Go back to the **Providers** tab described in step 1. If there is a DefaultAuthenticator entry there, click it, and set the control flag to SUFFICIENT and click **Save**.

5.14.1 Add an LDAP Authenticator (Setting Up Roles)

To add an LDAPAuthenticator:

1. In the WLS Admin Console go to your Security Realm and in the **Providers** tab, add a new LDAPAuthenticator.
2. On **Common** tab, set the Control Flag to SUFFICIENT.
3. Click the **Provider Specific** tab.
4. Set the host and the port of the OID server.
5. Set Pricipal to "orclApplicationCommonName=WLCSInstance1,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com" (replace domain part with your domain).
6. Set the Credential to what you configured in an earlier step and Confirm Credential.
7. Set User Base DN to "cn=Users,dc=example,dc=com" (replace domain part with your domain).
8. Enable Use Retrieved User Name as Principal.
9. Set Group Base DN to "cn=Groups,example,dc=com" (replace domain part with your domain).
10. Click **Save**.

5.14.2 Improving LDAP Authenticator Performance

You can improve LDAP Authenticator performance:

- If users are in a flat structure (which is usually the case), set User Search Scope to "onelevel".
- If the groups for roles are in a flat structure (which is also usually the case), that is, there are no groups in groups, set Group Search Scope to "onelevel", and Set Group Membership Searching to "limited".

Ensure that you save your changes.

5.14.3 Configuring Userservice to work with OID

You must configure *Userservice* to work with OID:

1. The ejb-jar.xml packaged in the subscriberdataservices ear file (subscriberdataservices-11.1.1.1.0.ear at $MW_HOME/as11gr1wlcs1/communications/applications)has to be configured to use LDAP instead of jdbc.
2. Extract the files in the subscriberdataservices ear followed by the files in the userservice jar file.
3. In ejb-jar.xml under META-INF, comment out the UserServiceDSN, and UserDAOImpl groups. Uncomment the LDAP section, and configure with proper values.

   Set the java.naming.security.principal to orclApplicationCommonName=WLCSInstance1,cn=WLCS,cn=Products,cn=OracleContext,dc=example,dc=com (replace domain part with your domain)

Set the password defined earlier - for plain text password prepended with a "!". For instance, to set the password to myPassword, configure <env-entry-value><![CDATA[!myPassword]]></env-entry-value>.

Set the provider LDAP URL. The default port is 389 (non SSL connections).

4. Save the ejb-jar.xml file with the above changes, repackage the jar file first followed by the ear file. Replace the existing subscriberdataservices ear file under $MW_HOME/as11gr1wlcs1/communications/applications with the modified one.
5. Restart OWLCS. You should now be able to successfully sign in to OWCLS as test.user1 using Oracle Communicator.

6

Configuring SIP Data Tier Partitions and Replicas

The following sections describe how to configure Oracle WebLogic Communication Services instances that make up the SIP data tier cluster of a deployment:

6.1 Overview of SIP Data Tier Configuration

The Oracle WebLogic Communication Services SIP data tier is a cluster of server instances that manages the application call state for concurrent SIP calls. The SIP data tier may manage a single copy of the call state or multiple copies as needed to ensure that call state data is not lost if a server machine fails or network connections are interrupted.

The SIP data tier cluster is arranged into one or more *partitions*. A partition consists of one or more SIP data tier server instances that manage the same portion of concurrent call state data. In a single-server Oracle WebLogic Communication Services installation, or in a two-server installation where one server resides in the engine tier and one resides in the SIP data tier, all call state data is maintained in a single partition. Multiple partitions are required when the size of the concurrent call state exceeds the maximum size that can be managed by a single server instance. When more than one partition is used, the concurrent call state is split among the partitions, and each partition manages an separate portion of the data. For example, with a two-partition SIP data tier, one partition manages the call state for half of the concurrent calls (for example, calls A through M) while the second partition manages the remaining calls (N through Z).

In most cases, the maximum call state size that can be managed by an individual server corresponds to the Java Virtual Machine limit of approximately 1.6GB per server.

Additional servers can be added within the same partition to manage copies of the call state data. When multiple servers are members of the same partition, each server manages a copy of the same portion of the call data, referred to as a *replica* of the call state. If a server in a partition fails or cannot be contacted due to a network failure, another replica in the partition supplies the call state data to the engine tier. Oracle recommends configuring two servers in each partition for production installations, to guard against machine or network failures. A partition can have a maximum of three replicas for providing additional redundancy.

6.1.1 datatier.xml Configuration File

The `datatier.xml` configuration file, located in the `config/custom` subdirectory of the domain directory, identifies SIP data tier servers and also defines the partitions and replicas used to manage the call state. If a server's name is present in `datatier.xml`, that server loads Oracle WebLogic Communication Services SIP data tier functionality at boot time. (Server names that do not appear in `datatier.xml` act as engine tier nodes, and instead provide SIP Servlet container functionality configured by the `sipserver.xml` configuration file.)

The sections that follow show examples of the `datatier.xml` contents for common SIP data tier configurations.

6.1.2 Configuration Requirements and Restrictions

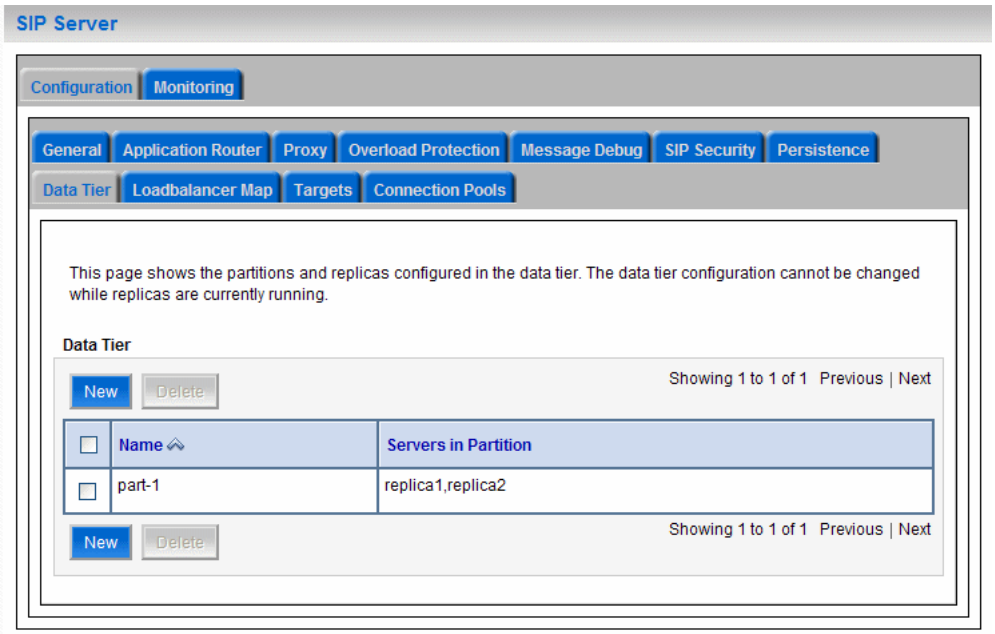
All servers that participate in the SIP data tier should be members of the same WebLogic Server cluster. The cluster configuration enables each server to monitor the status of other servers. Using a cluster also enables you to easily target the `sipserver` and `datatier` custom resources to all servers for deployment.

For high reliability, you can configure up to three replicas within a partition.

You cannot change the SIP data tier configuration while replicas or engine tier nodes are running. You must restart servers in the domain in order to change SIP data tier membership or reconfigure partitions or replicas.

You can view the current SIP data tier configuration (and configure the data tier) using the Configuration > Data Tier page (SipServer node) of the Administration Console, as shown in Figure 6–1.

Figure 6–1 Administration Console Display of SIP Data Tier Configuration (Read-Only)



6.2 Best Practices for Configuring and Managing SIP Data Tier Servers

Adding replicas can increase reliability for the system as a whole, but keep in mind that each additional server in a partition requires additional network bandwidth to manage the replicated data. With three replicas in a partition, each transaction that modifies the call state updates data on three different servers.

To ensure high reliability when using replicas, always ensure that server instances in the same partition reside on different machines. Hosting two or more replicas on the same machine leaves all of the hosted replicas vulnerable to a machine or network failure.

SIP data tier servers can have one of three different statuses:

- `ONLINE`—indicates that the server is available for managing call state transactions.
- `OFFLINE`—indicates that the server is shut down or unavailable.
- `ONLINE_LOCK_AUTHORITY_ONLY`—indicates that the server was rebooted and is currently being updated (from other replicas) with the current call state data. A recovering server cannot yet process call state transactions, because it does not maintain a full copy of the call state managed by the partition.

If you need to take a SIP data tier server instance offline for scheduled maintenance, make sure that at least one other server in the same partition is `active`. If you shut down an `active` server and all other servers in the partition are `offline` or `recovering`, you will lose a portion of the active call state.

Oracle WebLogic Communication Services automatically divides the call state evenly over all configured partitions.

6.3 Example SIP Data Tier Configurations and Configuration Files

The sections that follow describe some common Oracle WebLogic Communication Services installations that utilize a separate SIP data tier.

6.3.1 SIP Data Tier with One Partition

A single-partition, single-server SIP data tier represents the simplest data tier configuration. Example 6–1 shows a SIP data tier configuration for a single-server deployment.

Example 6–1 SIP Data Tier Configuration for Small Deployment

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http://www.bea.com/ns/wlcp/wlss/300">
    <partition>
      <name>part-1</name>
      <server-name>replica1</server-name>
    </partition>
  </data-tier>
```

To add a replica to an existing partition, simply define a second `server-name` entry in the same partition. For example, the `datatier.xml` configuration file shown in Example 6–2 recreates a two-replica configuration.

Example 6–2 SIP Data Tier Configuration for Small Deployment with Replication

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http://www.bea.com/ns/wlcp/wlss/300">
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
      <server-name>DataNode0-1</server-name>
    </partition>
  </data-tier>
```

6.3.2 SIP Data Tier with Two Partitions

Multiple partitions can be easily created by defining multiple `partition` entries in `datatier.xml`, as shown in Example 6–3.

Example 6–3 Two-Partition SIP Data Tier Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http://www.bea.com/ns/wlcp/wlss/300">
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
    </partition>
    <partition>
      <name>Partition1</name>
      <server-name>DataNode1-0</server-name>
    </partition>
  </data-tier>
```

6.3.3 SIP Data Tier with Two Partitions and Two Replicas

Replicas of the call state can be added by defining multiple SIP data tier servers in each partition. Example 6–4 shows the `datatier.xml` configuration file used to define a system having two partitions with two servers (replicas) in each partition.

Example 6–4 SIP Data Tier Configuration for Small Deployment

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http://www.bea.com/ns/wlcp/wlss/300">
```

```
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
      <server-name>DataNode0-1</server-name>
    </partition>
    <partition>
      <name>Partition1</name>
      <server-name>DataNode1-0</server-name>
      <server-name>DataNode1-1</server-name>
    </partition>
  </data-tier>
```

6.4 Storing Long-Lived Call State Data In A RDBMS

Oracle WebLogic Communication Services enables you to store long-lived call state data in an Oracle or MySQL RDBMS in order to conserve RAM. When you enable RDBMS persistence, by default the SIP data tier persists a call state's data to the RDBMS after the call dialog has been established, and at subsequent dialog boundaries, retrieving or deleting the persisted call state data as necessary to modify or remove the call state.

Oracle also provides an API for application designers to provide "hints" as to when the SIP data tier should persist call state data. These hints can be used to persist call state data to the RDBMS more frequently, or to disable persistence for certain calls.

Note that Oracle WebLogic Communication Services only uses the RDBMS to supplement the SIP data tier's in-memory replication functionality. To improve latency performance when using an RDBMS, the SIP data tier maintains SIP timers in memory, along with call states being actively modified (for example, in response to a new call being set up). Call states are automatically persisted only after a dialog has been established and a call is in progress, at subsequent dialog boundaries, or in response to persistence hints added by the application developer.

When used in conjunction with an RDBMS, the SIP data tier selects one replica server instance to process all call state writes (or deletes) to the database. Any available replica can be used to retrieve call states from the persistent store as necessary for subsequent reads.

RDBMS call state storage can be used in combination with an engine tier cache, if your domain uses a SIP-aware load balancer to manage connections to the engine tier. See Section 6.7, "Caching SIP Data in the Engine Tier".

6.4.1 Requirements and Restrictions

Enable RDBMS call state storage only when all of the following criteria are met:

- The call states managed by your system are typically long-lived.
- The size of the call state to be stored is large. Very large call states may require a significant amount of RAM in order to store the call state.
- Latency performance is not critical to your deployed applications.

The latency requirement, in particular, must be well understood before choosing to store call state data in an RDBMS. The RDBMS call state storage option measurably increases latency for SIP message processing, as compared to using a SIP data tier cluster. If your system must handle a large number of short-lived SIP transactions with brief response times, Oracle recommends storing all call state data in the SIP data tier.

Note: RDBMS persistence is designed only to reduce the RAM requirements in the SIP data tier for large, long-lived call states. The persisted data cannot be used to restore a failed SIP data tier partition or replica.

6.4.2 Steps for Enabling RDBMS Call State Storage

In order to use the RDBMS call state storage feature, your Oracle WebLogic Communication Services domain must include the necessary JDBC configuration, SIP Servlet container configuration, and a database having the schema required to store the call state. You can automate much of the required configuration by using the Configuration Wizard to set up a new domain with the RDBMS call state template. See Section 6.4.3, "Using the Configuration Wizard RDBMS Store Template".

If you have an existing Oracle WebLogic Communication Services domain, or you want to configure the RDBMS store on your own, see Section 6.4.4, "Configuring RDBMS Call State Storage by Hand" for instructions to configure JDBC and Oracle WebLogic Communication Services to use an RDBMS store.

6.4.3 Using the Configuration Wizard RDBMS Store Template

The Configuration Wizard provides a simple template that helps you easily begin using and testing the RDBMS call state store. Follow these steps to create a new domain from the template:

1. Start the Configuration Wizard application (config.sh)
2. Accept the default selection, *Create a new WebLogic domain*, and click **Next**.
3. Select *Base this domain on an existing template*, and click **Browse** to display the Select a Template dialog.
4. Select the template named `replicateddomain.jar`, and click **OK**.
5. Click **Next**.
6. Enter the username and password for the Administrator of the new domain, and click Next.
7. Select a JDK to use, and click Next.
8. Select No to keep the settings defined in the source template file, and click Next.
9. Click Create to create the domain.

   The template creates a new domain with two engine tier servers in a cluster, two SIP data tier servers in a cluster, and an Administration Server (AdminServer). The engine tier cluster includes the following resources and configuration:

   - A JDBC datasource, `wlss.callstate.datasource`, required for storing long-lived call state data. Note that you must modify this configuration to configure the datasource for your own RDBMS server. See Section 6.4.3.1, "Modify the JDBC Datasource Connection Information".
   - A persistence configuration (shown in the SipServer node, Configuration > Persistence tab of the Administration Console) that defines default handling of persistence hints for both RDBMS and geographical redundancy.
10. Click **Done** to exit the configuration wizard.
11. Follow the steps under Section 6.4.3.1, "Modify the JDBC Datasource Connection Information" to create the necessary tables in your RDBMS.

12. Follow the steps under Section 6.4.4.3, "Create the Database Schema" to create the necessary tables in your RDBMS.

6.4.3.1 Modify the JDBC Datasource Connection Information

After installing the new domain, modify the template JDBC datasource to include connection information for your RDBMS server:

1. Use your browser to access the URL http://*address*:*port*/console where *address* is the Administration Server's listen address and *port* is the listen port.
2. Select the Services > JDBC > Data Sources tab in the left pane.
3. Select the data source named wlss.callstate.datasource in the right pane.
4. Select the Configuration > Connection Pool tab in the right pane.
5. Modify the following connection pool properties:
   - **URL**: Modify the URL to specify the host name and port number of your RDBMS server.
   - **Properties**: Modify the value of the user, portNumber, SID, and serverName properties to match the connection information for your RDBMS.
   - **Password** and **Confirm Password**: Enter the password of the RDBMS user you specified.
6. Click **Save** to save your changes.
7. Select the **Targets** tab in the right pane.
8. On the Select Targets page, select the name of your SIP data tier cluster (for example, BEA_DATA_TIER_CLUST), then click **Save**.
9. Click **Save**.
10. Follow the steps under Section 6.4.4.3, "Create the Database Schema" to create the necessary tables in your RDBMS.

6.4.4 Configuring RDBMS Call State Storage by Hand

To change an existing Oracle WebLogic Communication Services domain to store call state data in an Oracle or MySQL RDBMS, you must configure the required JDBC datasource, edit the Oracle WebLogic Communication Services configuration, and add the required schema to your database. Follow the instructions in the sections below to configure an Oracle Database.

6.4.4.1 Configure JDBC Resources

Follow these steps to create the required JDBC resources in your domain:

1. Boot the Administration Server for the domain if it is not already running.
2. Access the Administration Console for the domain.
3. Select the Services > JDBC > Data Sources tab in the left pane.
4. Click New to create a new data source.
5. Fill in the fields of the Create a New JDBC Data Source page as follows:
   - **Name:** Enter wlss.callstate.datasource
   - **JNDI Name**: Enter wlss.callstate.datasource.
   - **Database Type**: Select "Oracle."

- **Database Driver**: Select an appropriate JDBC driver from the Database Driver list. Note that some of the drivers listed in this field may not be installed by default on your system. Install third-party drivers as necessary using the instructions from your RDBMS vendor.

6. Click **Next**.
7. Fill in the fields of the Connection Properties tab using connection information for the database you wan to use. Click Next to continue.
8. Click Test Configuration to test your connection to the RDBMS, or click Next to continue.
9. On the Select Targets page, select the name of your SIP data tier cluster (for example, BEA_DATA_TIER_CLUST).
10. Click Finish to save your changes.

6.4.4.2 Configure Oracle WebLogic Communication Services Persistence Options

Follow these steps to configure the Oracle WebLogic Communication Services persistence options to use an RDBMS call state store:

1. Boot the Administration Server for the domain if it is not already running.
2. Access the Administration Console for the domain.
3. Select the SipServer node in the left pane.
4. Select the Configuration > Persistence tab in the right pane.
5. In the Default Handling drop-down menu, select either "db" or "all." It is acceptable to select "all" because geographically-redundant replication is only performed if the Geo Site ID and Geo Remote T3 URL fields have been configured.
6. Click **Save** to save your changes.

6.4.4.3 Create the Database Schema

Oracle WebLogic Communication Services includes a SQL script, `callstate.sql`, that you can use to create the tables necessary for storing call state information. The script is installed to the `user_staged_config` subdirectory of the domain directory when you configure a replicated domain using the Configuration Wizard. The script is also available in the *WLSS_HOME*`/common/templates/scripts/db/oracle` directory.

The contents of the `callstate.sql` SQL script are shown in Example 6–5.

Example 6–5 callstate.sql Script for Call State Storage Schema

```
drop table callstate;

create table callstate (
  key1 int,
  key2 int,
  bytes blob default empty_blob(),
  constraint pk_callstate primary key (key1, key2)
);
```

Follow these steps to execute the script commands using SQL*Plus:

1. Move to the Oracle WebLogic Communication Services `utils` directory, in which the SQL Script is stored:

   ```
   cd ~/bea/wlcserver_10.3/common/templates/scripts/db/oracle
   ```

2. Start the SQL*Plus application, connecting to the Oracle database in which you will create the required tables. Use the same username, password, and connect to the same database that you specified when configuring the JDBC driver in Section 6.4.4.1, "Configure JDBC Resources". For example:

   ```
   sqlplus username/password@connect_identifier
   ```

   where `connect_identifier` connects to the database identified in the JDBC connection pool.

3. Execute the Oracle WebLogic Communication Services SQL script, `callstate.sql`:

   ```
   START callstate.sql
   ```

4. Exit SQL*Plus:

   ```
   EXIT
   ```

6.4.5 Using Persistence Hints in SIP Applications

Oracle WebLogic Communication Services provides a simple API to provide "hints" as to when the SIP data tier should persist call state data. You can use the API to disable persistence for specific calls or SIP requests, or to persist data more frequently than the default setting (at SIP dialog boundaries).

To use the API, simply obtain a `WlssSipApplicationSession` instance and use the `setPersist` method to enable or disable persistence. Note that you can enable or disable persistence either to an RDBMS store, or to as geographically-redundant Oracle WebLogic Communication Services installation (see Section 6.6, "Using Geographically-Redundant SIP Data Tiers").

For example, some SIP-aware load balancing products use the SIP OPTIONS message to determine if a SIP Server is active. To avoid persisting these messages to an RDBMS and to a geographically-redundant site, a Servlet might implement a `doOptions` method to echo the request and turn off persistence for the message, as shown in Example 6–6.

Example 6–6 Disabling RDBMS Persistence for Option Methods

```
protected void doOptions(SipServletRequest req) throws IOException {
    WlssSipApplicationSession session =
      (WlssSipApplicationSession) req.getApplicationSession();
    session.setPersist(WlssSipApplicationSession.PersistenceType.DATABASE,
      false);
    session.setPersist(WlssSipApplicationSession.PersistenceType.GEO_REDUNDANCY,
false);
    req.createResponse(200).send();
}
```

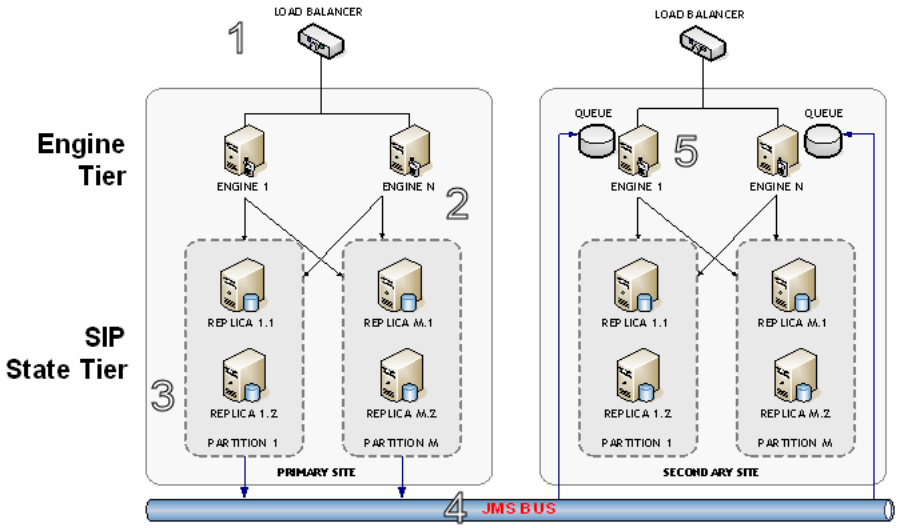
6.5 Introducing Geo-Redundancy

Geo-Redundancy ensures uninterrupted transactions and communications for providers, using geographically-separated SIP server deployments.

A primary site can process various SIP transactions and communications and upon determining a transaction boundary, replicate the state data associated with the transaction being processed, to a secondary site. Upon failure of the primary site, calls

are routed from the failed primary site to a secondary site for processing. Similarly, upon recovery, the calls are re-routed back to the primary site.

Figure 6–2 Geo-Redundancy



In the preceeding figure, Geo-Redundancy is portrayed. The process proceeds in this manner:

1. Call is initiated on a primary OCCAS Cluster site, call setup and processing occurs normally.
2. Call is replicated as usual to the site's SIP State Tier, and becomes eligible for replication to a secondary site.
3. A single replica in the SIP State Tier then places the call state data to be replicated on a JMS queue configured.
4. Call is transmitted to one of the available engines using JMS over WAN.
5. Engines at the secondary site monitor their local queue for new messages. Upon receiving a message, an Engine in the secondary site OCCAS Cluster persists the call state data and assigns it the site ID value of the primary site.

Table 6–1 Geographic Redundancy flow

Normal Operation	failover
When a session is initiated on a primary OCCAS site, call setup and processing occurs normally.	Global LB policy updated to begin routing calls - primary site to secondary site.
When a SIP transaction boundary is reached, the call is replicated (in-memory) to the site's data tier, and becomes eligible for replication to a secondary site.	Once complete, the secondary site begins processing requests for the backed-up call state data.
A single replica in the data tier then places the call state data to be replicated on a JMS queue configured on the replica site.	When a requests hit secondary site engine retrieves the data and activates the call state, taking ownership for the call.
Data is transmitted to one of the available engines round-robin fashion.	Sets the site ID associated with the call to zero (making it appear local).
Engines at the secondary site monitor their local queue for new messages.	Activates all dormant timers present in the call state.

Table 6–1 (Cont.) Geographic Redundancy flow

Normal Operation	failover
Upon receiving a message, an engine on the secondary site persists the call state data and assigns it the site ID value of the primary site.	By default, call states are activated only for individual calls, and only after those calls are requested on the backup site.
The site ID distinguishes replicated call state data on the secondary site from any other call state data actively managed by the secondary site.	Servlets can use the WlssSipApplicationSession.getGeoSiteId() method to examine the site ID associated with a call.
Timers in replicated call state data remain dormant on the secondary site, so that timer processing does not become a bottleneck to performance.	Any non-zero value for the site ID indicates that the Servlet is working with call state data that was replicated from another site.

6.5.1 Situations Best Suited to Use Geo-Redundancy

The following situations are best suited to take advantage of Geo-Redundancy:

- Your application uses SIP dialog states that are long-lived (dialog states that typically last 30 seconds or longer, such as SUBSCRIBE dialogs or conferences)
- Your application would reasonably be able to reconstruct the session (re-INVITE, expire SUBSCRIBE dialogs to trigger re-subscriptions, and so on) from the state that has been replicated
- The link between two OCCAS clusters or sites is low-bandwidth (<1Gb/s each direction) or high (or variable) latency (>5ms 95%)

6.5.2 Situations Not Suited to Use Geo-Redundancy

Geo-Redundancy should not be used in these situations:

- A high-capacity link between sites is available
- Your application does not reach SIP dialog steady-states that are likely to last longer than the time it would take to re-route all traffic to the secondary site in the event of catastrophic failure (15-30 seconds)
- If the application session is likely to be terminated by the user before the application could re-construct the session (most users will disconnect their calls before the session can be re-established from the secondary site)
- The volume of session state objects created by the application is greater than the site interconnect can support

6.5.3 Geo-Redundancy Considerations: Before Your Begin

Keep in mind the following considerations when planning your Geo-Redundancy:

- Dimension the system for the site link!
- Each dialog state is ~25KB on the wire (25600 bits)
- A typical B2BUA is two (2) dialogs
- Aim for 25% utilization (or less, depending on the specific equipment and topology of the site) to accommodate “jitter” and sustained latency on the link

  For example, a 100 Mb/s link can handle approximately1000 call states per second, and a typical B2BUA (in the default configuration) generates 4 states during the call (two for each dialog). So, a 100 Mb/s link will support a single OWLCS cluster dimensioned for a peak arrival rate (call rate) of 250 CPS.

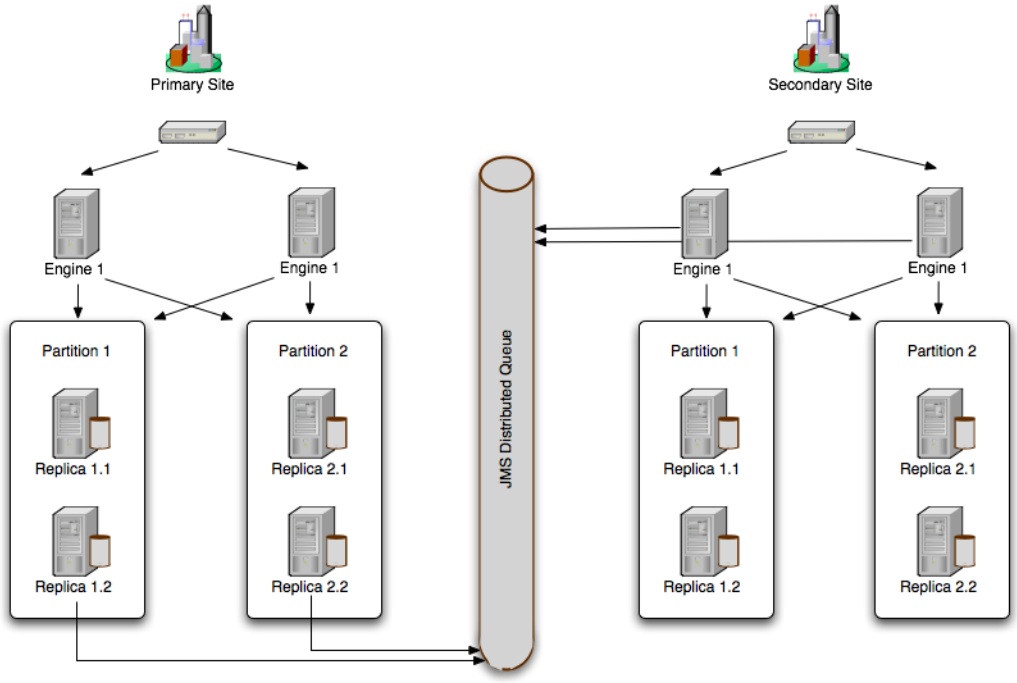
- Geo-Redundancy is not *transparent* to the application; in most cases the application must be designed to use `SetPersist()` appropriately, and the developer must consider the volume of state that the application will queue for replication between sites
- `SetPersist()` should be used within the application code to selectively identify dialog states that will be long-lived
- Given the time it generally takes to route traffic to a secondary site, any application that replicates state more frequently will unnecessarily saturate the JMS queue and site interconnect
- Tuning of JMS to the specific application environment is required: Serialization options, message batching, reliable delivery options and queue size are all variable, depending on the specific application and site characteristics
- Geo-Redundancy default behavior is to replicate all dialog state changes when Geo-Redundancy is enabled for the container (this is *not* recommended for production deployments)
- Given the time it generally takes to route traffic to a secondary site, any application that replicates state more frequently will unnecessarily saturate the site interconnect
- `SetPersist()` should be used within the application code to selectively identify dialog states that will be long-lived (longer than ~20-30 seconds would be a reasonable threshold)

6.6 Using Geographically-Redundant SIP Data Tiers

The basic call state replication functionality available in the Oracle WebLogic Communication Services SIP data tier provides excellent failover capabilities for a single site installation. However, the active replication performed within the SIP data tier requires high network bandwidth in order to meet the latency performance needs of most production networks. This bandwidth requirement makes a single SIP data tier cluster unsuitable for replicating data over large distances, such as from one regional data center to another.

The Oracle WebLogic Communication Services geographic persistence feature enables you to replica call state transactions across multiple Oracle WebLogic Communication Services installations (multiple Administrative domains or "sites"). A geographically-redundant configuration minimizes dropped calls in the event of a catastrophic failure of an entire site, for example due to an extended, regional power outage.

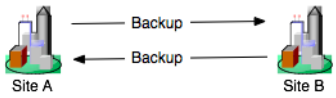
Figure 6–3 Oracle WebLogic Communication Services Geographic Persistence



6.6.1 Example Domain Configurations

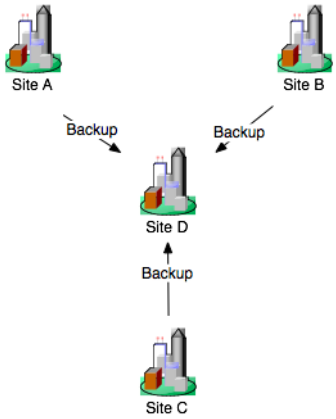
A secondary Oracle WebLogic Communication Services domain that persists data from another domain may itself process SIP traffic, or it may exist solely as an active standby domain. In the most common configuration, two sites are configured to replicate each other's call state data, with each site processing its own local SIP traffic. The administrator can then use either domain as the "secondary" site should one of domains fail.

Figure 6–4 Common Geographically-Redundant Configuration



An alternate configuration utilizes a single domain that persists data from multiple, other sites, acting as the secondary for those sites. Although the secondary site in this configuration can also process its own, local SIP traffic, keep in mind that the resource requirements of the site may be considerable because of the need to persist active traffic from several other installations.

Figure 6–5 Alternate Geographically-Redundant Configuration



When using geographic persistence, a single replica in the primary site places modified call state data on a distributed JMS queue. By default, data is placed on the queue only at SIP dialog boundaries. (A custom API is provided for application developers that want to replicate data using a finer granularity, as described in Section 6.4.5, "Using Persistence Hints in SIP Applications".) In a secondary site, engine tier servers use a message listener to monitor the distributed queue to receive messages and write the data to its own SIP data tier cluster. If the secondary site uses an RDBMS to store long-lived call states (recommended), then all data writes from the distribute queue go directly to the RDBMS, rather than to the in-memory storage of the SIP data tier.

6.6.2 Requirements and Limitations

The Oracle WebLogic Communication Services geographically-redundant persistence feature is most useful for sites that manage long-lived call state data in an RDBMS. Short-lived calls may be lost in the transition to a secondary site, because Oracle WebLogic Communication Services may choose to collect data for multiple call states before replicating between sites.

You must have a reliable, site-aware load balancing solution that can partition calls between geographic locations, as well as monitor the health of a given regional site. Oracle WebLogic Communication Services provides no automated functionality for detecting the failure of an entire domain, or for failing over to a secondary site. It is the responsibility of the Administrator to determine when a given site has "failed," and to redirect that site's calls to the correct secondary site. Furthermore, the site-aware load balancer must direct all messages for a given callId to a single home site (the "active" site). If, after a failover, the failed site is restored, the load balancer must continue directing calls to the active site and not partition calls between the two sites.

During a failover to a secondary site, some calls may be dropped. This can occur because Oracle WebLogic Communication Services generally queues call state data for site replication only at SIP dialog boundaries. Failures that occur before the data is written to the queue result in the loss of the queued data.

Also, Oracle WebLogic Communication Services replicates call state data across sites only when a SIP dialog boundary changes the call state. If a long-running call exists on the primary site before the secondary site is started, and the call state remains unmodified, that call's data is not replicated to the secondary site. Should a failure occur before a long-running call state has been replicated, the call is lost during failover.

When planning for the capacity of a Oracle WebLogic Communication Services installation, keep in mind that, after a failover, a given site must be able to support all of the calls from the failed site as well as from its own geographic location. Essentially this means that all sites that are involved in a geographically-redundant configuration will operate at less than maximum capacity until a failover occurs.

6.6.3 Steps for Configuring Geographic Persistence

In order to use the Oracle WebLogic Communication Services geographic persistence features, you must perform certain configuration tasks on both the primary "home" site and on the secondary replication site.

Table 6–2 Steps for Configuring Geographic Persistence

Steps for Primary "Home" Site	Steps for Secondary "Replication" Site:
1. Install Oracle WebLogic Communication Services software and create replicated domain.	1. Install Oracle WebLogic Communication Services software and create replicated domain.
2. Enable RDBMS storage for long-lived call states (recommended).	2. Enable RDBMS storage for long-lived call states (recommended).
3. Configure persistence options to: define the unique regional site ID; identify the secondary site's URL; and enable replication hints.	3. Configure JMS Servers and modules required for replicating data.
	4. Configure persistence options to define the unique regional site ID.

Note: In most production deployments, two sites will perform replication services for each other, so you will generally configure each installation as both a primary and secondary site.

Oracle WebLogic Communication Services provides domain templates to automate the configuration of most of the resources described in Table 6–2. See Section 6.6.4, "Using the Configuration Wizard Templates for Geographic Persistence" for information about using the templates.

If you have an existing Oracle WebLogic Communication Services domain and want to use geographic persistence, follow the instructions in Section 6.6.5, "Manually Configuring Geographical Redundancy" to create the resources.

6.6.4 Using the Configuration Wizard Templates for Geographic Persistence

Oracle WebLogic Communication Services provides two Configuration Wizard templates for using geographic persistence features:

- `WLSS_HOME/common/templates/domains/geo1domain.jar` configures a primary site having a site ID of 1. The domain replicates data to the engine tier servers created in `geo2domain.jar`.
- `WLSS_HOME/common/templates/domains/geo2domain.jar` configures a secondary site that replicates call state data from the domain created with `geo1domain.jar`. This installation has site ID of 2.

The server port numbers in both domain templates are unique, so you can test geographic persistence features on a single machine if necessary. Follow the instructions in the sections that follow to install and configure each domain.

6.6.4.1 Installing and Configuring the Primary Site

Follow these steps to create a new primary domain from the template:

1. Start the Configuration Wizard application (config.sh).
2. Accept the default selection, *Create a new WebLogic domain*, and click **Next**.
3. Select Base this domain on an existing template, and click Browse to display the Select a Template dialog.
4. Select the template named `geo1domain.jar`, and click **OK**.
5. Click **Next**.
6. Enter the username and password for the Administrator of the new domain, and click **Next**.
7. Select a JDK to use, and click **Next**.
8. Select No to keep the settings defined in the source template file, and click **Next**.
9. Click **Create** to create the domain.

   The template creates a new domain with two engine tier servers in a cluster, two SIP data tier servers in a cluster, and an Administration Server (AdminServer). The engine tier cluster includes the following resources and configuration:

   - A JDBC datasource, `wlss.callstate.datasource`, required for storing long-lived call state data. If you want to use this functionality, edit the datasource to include your RDBMS connection information as described in Section 6.4.3.1, "Modify the JDBC Datasource Connection Information".
   - A persistence configuration (shown in the SipServer node, Configuration > Persistence tab of the Administration Console) that defines:
     - Default handling of persistence hints for both RDBMS and geographic persistence.
     - A Geo Site ID of 1.
     - A Geo Remote T3 URL of `t3://localhost:8011,localhost:8061`, which identifies the engine tier servers in the "geo2" domain as the replication site for geographic redundancy.
10. Click Done to exit the configuration wizard.
11. Follow the steps under Section 6.6.4.2, "Installing the Secondary Site" to create the domain that performs the replication.

6.6.4.2 Installing the Secondary Site

Follow these steps to use a template to create a secondary site from replicating call state data from the "geo1" domain:

1. Start the Configuration Wizard application (config.sh).
2. Accept the default selection, *Create a new WebLogic domain*, and click **Next**.
3. Select Base this domain on an existing template, and click Browse to display the Select a Template dialog.
4. Select the template named `geo2domain.jar`, and click **OK**.
5. Click **Next**.
6. Enter the username and password for the Administrator of the new domain, and click **Next**.

7. Select a JDK to use, and click **Next**.
8. Select No to keep the settings defined in the source template file, and click **Next**.
9. Click **Create** to create the domain.

   The template creates a new domain with two engine tier servers in a cluster, two SIP data tier servers in a cluster, and an Administration Server (AdminServer). The engine tier cluster includes the following resources and configuration:

   - A JDBC datasource, `wlss.callstate.datasource`, required for storing long-lived call state data. If you want to use this functionality, edit the datasource to include your RDBMS connection information as described in Section 6.4.3.1, "Modify the JDBC Datasource Connection Information".
   - A persistence configuration (shown in the SipServer node, Configuration > Persistence tab of the Administration Console) that defines:
     - Default handling of persistence hints for both RDBMS and geographical redundancy.
     - A Geo Site ID of 2.
   - A JMS system module, `SystemModule-Callstate`, that includes:
     - `ConnectionFactory-Callstate`, a connection factory required for backing up call state data from a primary site.
     - `DistributedQueue-Callstate`, a uniform distributed queue required for backing up call state data from a primary site.

     The JMS system module is targeted to the site's engine tier cluster
   - Two JMS Servers, `JMSServer-1` and `JMSServer-2`, are deployed to `engine1-site2` and `engine2-site2`, respectively.
10. Click **Done** to exit the configuration wizard.

6.6.5 Manually Configuring Geographical Redundancy

If you have an existing replicated Oracle WebLogic Communication Services installation, or pair of installations, you must create by hand the JMS and JDBC resources required for enabling geographical redundancy. You must also configure each site to perform replication. These basic steps for enabling geographical redundancy are:

1. Configure JDBC Resources. Oracle recommends configuring both the primary and secondary sites to store long-lived call state data in an RDBMS.
2. Configure Persistence Options. Persistence options must be configured on both the primary and secondary sites to enable engine tier hints to write to an RDBMS or to replicate data to a geographically-redundant installation.
3. Configure JMS Resources. A secondary site must have available JMS Servers and specific JMS module resources in order to replicate call state data from another site.

The sections that follow describe each step in detail.

6.6.5.1 Configuring JDBC Resources (Primary and Secondary Sites)

Follow the instructions in Section 6.4, "Storing Long-Lived Call State Data In A RDBMS" to configure the JDBC resources required for storing long-lived call states in an RDBMS.

6.6.5.2 Configuring Persistence Options (Primary and Secondary Sites)

Both the primary and secondary sites must configure the correct persistence settings in order to enable replication for geographical redundancy. Follow these steps to configure persistence:

1. Use your browser to access the URL http://*address*:*port*/console where *address* is the Administration Server's listen address and *port* is the listen port.
2. Click **Lock & Edit** to obtain a configuration lock.
3. Select the SipServer node in the left pane. The right pane of the console provides two levels of tabbed pages that are used for configuring and monitoring Oracle WebLogic Communication Services.
4. Select the **Configuration > Persistence** tab in the right pane.
5. Configure the Persistence attributes as follows:
   - **Default Handling**: Select "all" to persist long-lived call state data to an RDBMS and to replicate data to an external site for geographical redundancy (recommended). If your installation does not store call state data in an RDBMS, select "geo" instead of "all."
   - **Geo Site ID**: Enter a unique number from 1 to 9 to distinguish this site from all other configured sites. Note that the site ID of 0 is reserved to indicate call states that are local to the site in question (call states not replicated from another site).
   - **Geo Remote T3 URL**: For primary sites (or for secondary sites that replicate their own data to another site), enter the T3 URL or URLs of the engine tier servers that will replicate this site's call state data. If the secondary engine tier cluster uses a cluster address, you can enter a single T3 URL, such as t3://mycluster:7001. If the secondary engine tier cluster does not use a cluster address, enter the URLs for each individual engine tier server separated by a comma, such as t3://engine1-east-coast:7001,t3://engine2-east-coast:7002,t3://engine3-east-coast:7001,t4://engine4-east-coast:7002.
6. Click **Save** to save your configuration changes.
7. Click **Activate Changes** to apply your changes to the engine tier servers.

6.6.5.3 Configuring JMS Resources (Secondary Site Only)

Any site that replicates call state data from another site must configure certain required JMS resources. The resources are not required for sites that do not replicate data from another site.

Follow these steps to configure JMS resources:

1. Use your browser to access the URL http://*address*:*port*/console where *address* is the Administration Server's listen address and *port* is the listen port.
2. Click **Lock & Edit** to obtain a configuration lock.
3. Select the **Services > Messaging > JMS Servers** tab in the left pane.
4. Click **New** in the right pane.
5. Enter a unique name for the JMS Server or accept the default name. Click **Next** to continue.
6. In the Target list, select the name of a single engine tier server node in the installation. Click **Finish** to create the new Server.

7. Repeat Steps 3-6 for to create a dedicated JMS Server for each engine tier server node in your installation.
8. Select the **Services > Messaging > JMS Modules** node in the left pane.
9. Click **New** in the right pane.
10. Fill in the fields of the Create JMS System Module page as follows:
    - **Name**: Enter a name for the new module, or accept the default name.
    - **Descriptor File Name**: Enter the prefix a configuration file name in which to store the JMS module configuration (for example, `systemmodule-callstate`).
11. Click **Next** to continue.
12. Select the name of the engine tier cluster, and choose the option **All servers in the cluster**.
13. Click Next to continue.
14. Select **Would you like to add resources to this JMS system module** and click Finish to create the module.
15. Click New to add a new resource to the module.
16. Select the **Connection Factory** option and click Next.
17. Fill in the fields of the Create a new JMS System Module Resource as follows:
    - **Name**: Enter a descriptive name for the resource, such as ConnectionFactory-Callstate.
    - **JNDI Name**: Enter the name `wlss.callstate.backup.site.connection.factory`.
18. Click Next to continue.
19. Click **Finish** to save the new resource.
20. Select the name of the connection factory resource you just created.
21. Select the **Configuration > Load Balance** tab in the right pane.
22. De-select the **Server Affinity Enabled** option, and click **Save**.
23. Re-select the **Services > Messaging > JMS Modules** node in the left pane.
24. Select the name of the JMS module you created in the right pane.
25. Click **New** to create another JMS resource.
26. Select the **Distributed Queue** option and click Next.
27. Fill in the **Name** field of the Create a new JMS System Module Resource by entering a descriptive name for the resource, such as DistributedQueue-Callstate.
28. **JNDI Name**: Enter the name Fill in the fields of the Create a new JMS System Module Resource as follows:
    - **Name**: Enter a descriptive name for the resource, such as ConnectionFactory-Callstate.
    - **JNDI Name**: Enter the name `wlss.callstate.backup.site.queue`.
29. Click Next to continue.
30. Click Finish to save the new resource.

31. Click **Save** to save your configuration changes.
32. Click **Activate Changes** to apply your changes to the engine tier servers.

6.6.6 Understanding Geo-Redundant Replication Behavior

This section provides more detail into how multiple sites replicate call state data. Administrators can use this information to better understand the mechanics of geo-redundant replication and to better troubleshoot any problems that may occur in such a configuration. Note, however, that the internal workings of replication across Oracle WebLogic Communication Services installations is subject to change in future releases of the product.

6.6.6.1 Call State Replication Process

When a call is initiated on a primary Oracle WebLogic Communication Services site, call setup and processing occurs normally. When a SIP dialog boundary is reached, the call is replicated (in-memory) to the site's SIP data tier, and becomes eligible for replication to a secondary site. Oracle WebLogic Communication Services may choose to aggregate multiple call states for replication in order to optimize network usage.

A single replica in the SIP data tier then places the call state data to be replicated on a JMS queue configured on the replica site. Data is transmitted to one of the available engines (specified in the `geo-remote-t3-url` element in `sipserver.xml`) in a round-robin fashion. Engines at the secondary site monitor their local queue for new messages.

Upon receiving a message, an engine on the secondary site persists the call state data and assigns it the site ID value of the primary site. The site ID distinguishes replicated call state data on the secondary site from any other call state data actively managed by the secondary site. Timers in replicated call state data remain dormant on the secondary site, so that timer processing does not become a bottleneck to performance.

6.6.6.2 Call State Processing After Failover

To perform a failover, the Administrator must change a global load balancer policy to begin routing calls from the primary, failed site to the secondary site. After this process is completed, the secondary site begins processing requests for the backed-up call state data. When a request is made for data that has been replicated from the failed site, the engine retrieves the data and activates the call state, taking ownership for the call. The activation process involves:

- Setting the site ID associated with the call to zero (making it appear local).
- Activating all dormant timers present in the call state.

By default, call states are activated only for individual calls, and only after those calls are requested on the backup site. `SipServerRuntimeMBean` includes a method, `activateBackup(byte site)`, that can be used to force a site to take over all call state data that it has replicated from another site. The Administrator can execute this method using a WLST configuration script. Alternatively, an application deployed on the server can detect when a request for replicated site data occurs, and then execute the method. Example 6–7 shows sample code from a JSP that activates a secondary site, changing ownership of all call state data replicated from site 1. Similar code could be used within a deployed Servlet. Note that either a JSP or Servlet must run as a privileged user in order to execute the `activateBackup` method.

In order to detect whether a particular call state request, Servlets can use the `WlssSipApplicationSession.getGeoSiteId()` method to examine the site ID

associated with a call. Any non-zero value for the site ID indicates that the Servlet is working with call state data that was replicated from another site.

Example 6–7 Activating a Secondary Site Using JMX

```
<%
    byte site = 1;

    InitialContext ctx = new InitialContext();
    MBeanServer server = (MBeanServer) ctx.lookup("java:comp/env/jmx/runtime");
    Set set = server.queryMBeans(new ObjectName("*:*,Type=SipServerRuntime"),
null);
    if (set.size() == 0) {
      throw new IllegalStateException("No MBeans Found!!!");
    }

    ObjectInstance oi = (ObjectInstance) set.iterator().next();
    SipServerRuntimeMBean bean = (SipServerRuntimeMBean)
      MBeanServerInvocationHandler.newProxyInstance(server,
        oi.getObjectName());

    bean.activateBackup(site);
  %>
```

Note that after a failover, the load balancer must route all calls having the same callId to the newly-activated site. Even if the original, failed site is restored to service, the load balancer must not partition calls between the two geographical sites.

6.6.7 Removing Backup Call States

You may also choose to stop replicating call states to a remote site in order to perform maintenance on the remote site or to change the backup site entirely. Replication can be stopped by setting the **Site Handling** attribute to "none" on the primary site as described in Section 6.6.5.2, "Configuring Persistence Options (Primary and Secondary Sites)".

After disabling geographical replication on the primary site, you also may want to remove backup call states on the secondary site. `SipServerRuntimeMBean` includes a method, `deleteBackup(byte site)`, that can be used to force a site to remove all call state data that it has replicated from another site. The Administrator can execute this method using a WLST configuration script or via an application deployed on the secondary site. The steps for executing this method are similar to those for using the `activateBackup` method, described in Section 6.6.6.2, "Call State Processing After Failover".

6.6.8 Monitoring Replication Across Regional Sites

The `ReplicaRuntimeMBean` includes two new methods to retrieve data about geographically-redundant replication:

- `getBackupStoreOutboundStatistics()` provides information about the number of calls queued to a secondary site's JMS queue.
- `getBackupStoreInboundStatistics()` provides information about the call state data that a secondary site replicates from another site.

See *Oracle Fusion Middleware Communication Services Java API Reference* for more information about `ReplicaRuntimeMBean`.

6.6.9 Troubleshooting Geographical Replication

In addition to using the `ReplicaRuntimeMBean` methods described in Section 6.6.8, "Monitoring Replication Across Regional Sites", Administrators should monitor any SNMP traps that indicate failed database writes on a secondary site installation.

Administrators must also ensure that all sites participating in geographically-redundant configurations use unique site IDs.

6.7 Caching SIP Data in the Engine Tier

As described in Chapter 15, "Oracle WebLogic Communication Services Base Platform Topologies", in the default Oracle WebLogic Communication Services configuration the engine tier cluster is stateless. A separate SIP data tier cluster manages call state data in one or more partitions, and engine tier servers fetch and write data in the SIP data tier as necessary. Engines can write call state data to multiple replicas in each partition to provide automatic failover should a SIP data tier replica going offline.

Oracle WebLogic Communication Services also provides the option for engine tier servers to cache a portion of the call state data locally, as well as in the SIP data tier. When a local cache is used, an engine tier server first checks its local cache for existing call state data. If the cache contains the required data, and the local copy of the data is up-to-date (compared to the SIP data tier copy), the engine locks the call state in the SIP data tier but reads directly from its cache. This improves response time performance for the request, because the engine does not have to retrieve the call state data from a SIP data tier server.

The engine tier cache stores only the call state data that has been most recently used by engine tier servers. Call state data is moved into an engine's local cache as necessary in order to respond to client requests or to refresh out-of-date data. If the cache is full when a new call state must be written to the cache, the least-recently accessed call state entry is first removed from the cache. The size of the engine tier cache is not configurable.

Using a local cache is most beneficial when a SIP-aware load balancer manages requests to the engine tier cluster. With a SIP-aware load balancer, all of the requests for an established call are directed to the same engine tier server, which improves the effectiveness of the cache. If you do not use a SIP-aware load balancer, the effectiveness of the cache is limited, because subsequent requests for the same call may be distributed to different engine tier severs (having different cache contents).

6.7.1 Configuring Engine Tier Caching

Engine tier caching is enabled by default. To disable partial caching of call state data in the engine tier, specify the `engine-call-state-cache-enabled` element in `sipserver.xml`:

```
<engine-call-state-cache-enabled>false</engine-call-state-cache-enabled>
```

When enabled, the cache size is fixed at a maximum of 250 call states. The size of the engine tier cache is not configurable.

6.7.2 Monitoring and Tuning Cache Performance

`SipPerformanceRuntime` monitors the behavior of the engine tier cache. Table 6–3 describes the MBean attributes.

Table 6–3 SipPerformanceRuntime Attribute Summary

Attribute	Description
`cacheRequests`	Tracks the total number of requests for session data items.
`cacheHits`	The server increments this attribute each time a request for session data results in a version of that data being found in the engine tier server's local cache. Note that this counter is incremented even if the cached data is out-of-date and needs to be updated with data from the SIP data tier.
`cacheValidHits`	This attribute is incremented each time a request for session data is fully satisfied by a cached version of the data.

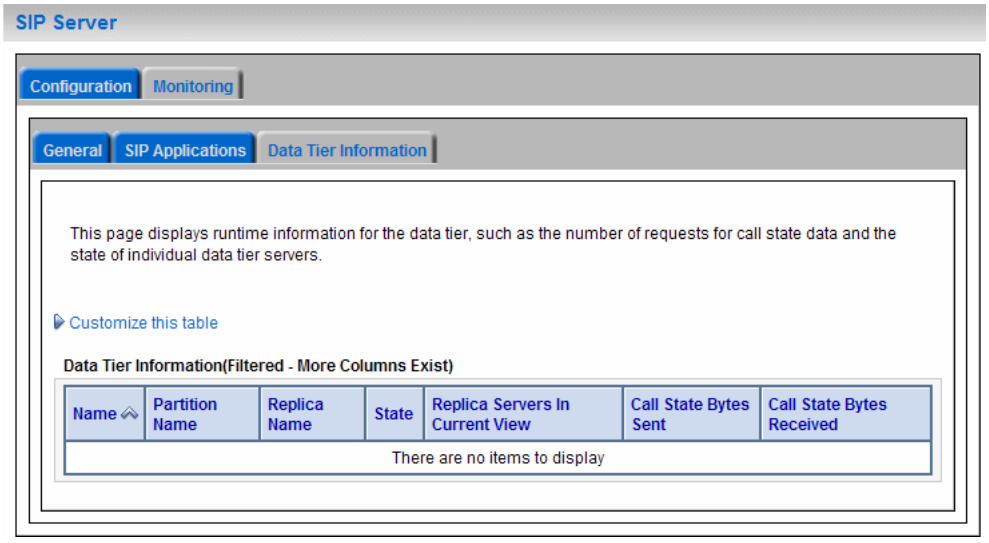
When enabled, the size of the cache is fixed at 250 call states. Because the cache consumes memory, you may need to modify the JVM settings used to run engine tier servers to meet your performance goals. Cached call states are maintained in the tenured store of the garbage collector. Try reducing the fixed "NewSize" value when the cache is enabled (for example, `-XX:MaxNewSize=32m -XX:NewSize=32m`). Note that the actual value depends on the call state size used by applications, as well as the size of the applications themselves.

6.8 Monitoring and Troubleshooting SIP Data Tier Servers

A runtime MBean, (`ReplicaRuntimeMBean`), provides valuable information about the current state and configuration of the SIP data tier. See *Oracle Fusion Middleware Communication Services Java API Reference* for a description of the attributes provided in this MBean.

Many of these attributes can be viewed using the SIP Servers Monitoring > Data Tier Information tab in the Administration Console, as shown in Figure 6–6.

Figure 6–6 SIP Data Tier Monitoring in the Administration Console



Example 6–8 shows a simple WLST session that queries the current attributes of a single Managed Server instance in a SIP data tier partition. Table 6–1 describes the MBean services in more detail.

Example 6–8 Displaying ReplicaRuntimeMBean Attributes

```
connect('weblogic','weblogic','t3://datahost1:7001')
custom()
cd('com.bea')
cd('com.bea:ServerRuntime=replica1,Name=replica1,Type=ReplicaRuntime')
ls()
-rw-   BackupStoreInboundStatistics               null
-rw-   BackupStoreOutboundStatistics              null
-rw-   BytesReceived                              0
-rw-   BytesSent                                  0
-rw-   CurrentViewId                              2
-rw-   DataItemCount                              0
-rw-   DataItemsToRecover                         0
-rw-   DatabaseStoreStatistics                    null
-rw-   HighKeyCount                               0
-rw-   HighTotalBytes                             0
-rw-   KeyCount                                   0
-rw-   Name                                       replica1
-rw-   Parent                                     com.bea:Name=replica1,Type=S
erverRuntime
-rw-   PartitionId                                0
-rw-   PartitionName                              part-1
-rw-   ReplicaId                                  0
-rw-   ReplicaName                                replica1
-rw-   ReplicaServersInCurrentView                java.lang.String[replica1,
replica2]
-rw-   ReplicasInCurrentView                      [I@75378c
-rw-   State                                      ONLINE
-rw-   TimerQueueSize                             0
-rw-   TotalBytes                                 0
-rw-   Type                                       ReplicaRuntime
```

Table 6–4 ReplicaRuntimeMBean Method and Attribute Summary

Method/Attribute	Description
`dumpState()`	Records the entire state of the selected SIP data tier server instance to the Oracle WebLogic Communication Services log file. You may want to use the `dumpState()` method to provide additional diagnostic information to a Technical Support representative in the event of a problem.
`BackupStoreInboundStatistics`	Provides statistics about call state data replicated from a remote geographical site.
`BackupStoreOutboundStatistics`	Provides statistics about call state data replicated to a remote geographical site.
`BytesReceived`	The total number of bytes received by this SIP data tier server. Bytes are received as servers in the engine tier provide call state data to be stored.
`BytesSent`	The total number of bytes sent from this SIP data tier server. Bytes are sent to engine tier servers when requested to provide the stored call state.

Table 6–4 (Cont.) ReplicaRuntimeMBean Method and Attribute Summary

Method/Attribute	Description
CurrentViewId	The current view ID. Each time the layout of the SIP data tier changes, the view ID is incremented. For example, as multiple servers in a SIP data tier cluster are started for the first time, the view ID is incremented when each server begins participating in the SIP data tier. Similarly, the view is incremented if a server is removed from the SIP data tier, either intentionally or due to a failure.
DataItemCount	The total number of stored call state keys for which this server has data. This attribute may be lower than the KeyCount attribute if the server is currently recovering data.
DataItemsToRecover	The total number of call state keys that must still be recovered from other replicas in the partition. A SIP data tier server may recover keys when it has been taken offline for maintenance and is then restarted to join the partition.
HighKeyCount	The highest total number of call state keys that have been managed by this server since the server was started.
HighTotalBytes	The highest total number of bytes occupied by call state data that this server has managed since the server was started.
KeyCount	The number of call data keys that are stored on the replica.
PartitionId	The numerical partition ID (from 0 to 7) of this server's partition.
PartitionName	The name of this server's partition.
ReplicaId	The numerical replica ID (from 0 to 2) of this server's replica.
ReplicaName	The name of this server's replica.
ReplicaServersInCurrentView	The names of other Oracle WebLogic Communication Services instances that are participating in the partition.
State	The current state of the replica. SIP data tier servers can have one of three different statuses: ■ ONLINE—indicates that the server is available for managing call state transactions. ■ OFFLINE—indicates that the server is shut down or unavailable. ■ ONLINE_LOCK_AUTHORITY_ONLY—indicates that the server was rebooted and is currently being updated (from other replicas) with the current call state data. A recovering server cannot yet process call state transactions, because it does not maintain a full copy of the call state managed by the partition.

Table 6–4 (Cont.) ReplicaRuntimeMBean Method and Attribute Summary

Method/Attribute	Description
`TimerQueueSize`	The current number of timers queued on the SIP data tier server. This generally corresponds to the KeyCount value, but may be less if new call states are being added but their associated timers have not yet been queued. **Note:** Engine tier servers periodically check with SIP data tier instances to determine if timers associated with a call have expired. In order for SIP timers to function properly, all engine tier servers must actively synchronize their system clocks to a common time source. Oracle recommends using a Network Time Protocol (NTP) client or daemon on each engine tier instance and synchronizing to a selected NTP server. See Section 3.5, "Configuring Timer Processing".
`TotalBytes`	The total number of bytes consumed by the call state managed in this server.

7

Provisioning Users With Sash

This chapter describes using the Sash utility. This chapter includes the following sections:

7.1 Overview of Sash

The Sapphire Shell (Sash) is a command-line utility to provision OWLCS users to the Oracle database, the XDMS (XML Document Management Server) and the RADIUS server. You can provision users from the Sash command line prompt (`sash#`) or by using the CommandService MBean.

See *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* for information on using Oracle Internet Database (OID) as the user provisioning repository for an OWLCS deployment.

7.2 Launching Sash

On Linux systems, the Sash launcher script (`sash.sh`) is located in the same folder that contains the start and stop scripts for OWLCS.

7.2.1 Launching Sash from the Command Line

OWLCS provides the following shortcuts for launching Sash from the command line:

`sash.sh` (UNIX)

`sash.bat` (Windows)

This shortcut is located at Oracle/Middleware/as11gr1wlcs1/communications/sash/bin on OWLCS installations.

7.2.2 Connecting Sash to an External OWLCS Instance

By default, Sash connects to the local instance of OWLCS. If needed, you can override this default behavior and connect Sash to external instances of OWLCS or to another instance of Oracle WebLogic Server.

7.2.2.1 Connecting to an External Instance of OWLCS

Sash connects to the OWLCS server through RMI. Example 7–1 illustrates how to connect Sash to a server with the host IP address 10.0.0.234.

Example 7–1 Connecting Sash to OWLCS

```
sash --host 10.0.0.234
```

When you connect to OWLCS, Sash prompts you for a username and a password. The user name is the same as that for OWLCS administrator (OWLCSadmin). The password is the same as the password associated with the OWLCS administrator. Once you log in, the Sash command prompt (`sash`) appears. An error message displays if the login is unsuccessful.

7.3 Using Sash

There are two groups of Sash commands: commands that create, delete and update system objects and commands that query the system for information.

Note: Whenever a user adds a new application usage, they must restart the server before the new application usage is available.

Whenever a user deletes an existing application usage, they must restart the server for the deleted application usage to be completely unloaded (that is, a deleted application usage will remain loaded until the server is restarted, when it is unloaded and is then completely unavailable).

If a space precedes a sash command in a file, and then that file is used as input to the sash command, it does not work. Ensure that you remove any preceding spaces in sash commands in sash input files.

7.3.1 Viewing Available Commands

Entering `help` displays a list of all available commands in the server (described in Table 7–1). The list of commands varies depending on the components deployed to the server.

Table 7–1 Sash Commands

Command	Description	Aliases	Subcommands

`privateIdentity`	Commands for adding and removing private communication identities used for authentication.	None	Subcommands include:
			■ `add` – Adds a new user to the system. For example:
			`privateIdentity add privateId=alice`
			■ `delete` – Removes a user from the system. For example:
			`privateIdentity delete privateId=alice`

`publicIdentity`	Commands for adding and removing public identities associated with a private identity.	`pubid`	Subcommands include:
			■ `add` – Adds a public identity to the system which is associated with a particular user. For example:
			`publicIdentity add publicId=sip:alice@test.company.com privateId=alice`
			■ `delete` – Deletes a communication identity from the system. For example:
			`publicIdentity delete publicId=sip:alice@test.company.com privateId=alice`

`account`	Contains commands for managing user accounts. This command enables you to set the account as active, locked, or as a temporary account.	None	Subcommands include:
			■ `add` – adds a new account to the system. The syntax is as follows:
			`account add uid=<string> [active=<true|false>] [locked=<true|false>] [accountExpiresAt=<accountExpiresAt>] [tempAccount=<true|false>] [description=<string>] [lockExpiresAt=<lockExpiresAt>] [currentFailedLogins=<integer>]`
			For example: `account add uid=alice active=true`
			■ `delete` – Deletes an account from the system. For example: `account delete uid=<string>`
			■ `update` – Updates an account. For example:
			`account update uid=<string> [active=<true|false>] [locked=<true|false>] [accountExpiresAt=<accountExpiresAt>] [tempAccount=<true|false>] [description=<string>] [lockExpiresAt=<lockExpiresAt>] [currentFailedLogins=<integer>]`
			■ info – Retrieves information for a specific account. For example: `account info uid=<string>`

Table 7–1 (Cont.) Sash Commands

Command	Description	Aliases	Subcommands
`role`	Manages role types and user roles in the system. `role` is an additional security and authorization mechanism that is defined within the `<auth-constraint>` element of `sip.xml`. This command authorizes a group of users access to applications. The applications in turn check for a specific role. OWLCS defines one role for the Proxy Registrar application, "Location Services".	None	Subcommands include `role system` and `role user`.

`role system` (subcommand of `role`)	Manages the roles types.	None	Subcommands include:
			■ `list` – Lists the roles in the system. For example:
			`role system list`
			■ `add` – Adds a new role to the system. For example:
			`role system add name=<string> [description=<string>]`
			■ `update` – Updates a role in the system. For example:
			`role system update name=<string> [description=<string>]`
			■ `delete` – Deletes a role from the system. For example:
			`role system delete name=<string> [description=<string>]`

Table 7–1 (Cont.) Sash Commands

Command	Description	Aliases	Subcommands

`role user` (subcommand of `role`)	Manages the user roles	None	Subcommands include:
			▪ `add` – Adds a role to a user. For example:
			`role user add uid=<string> name=<string>`
			▪ `delete` – Deletes a role from a user. For example:
			`role user delete uid=<string> name=<string>`
			▪ `list` – Lists roles for a user. For example:
			`role user list uid=<string>`

credentials	Command for managing credentials.	None	Subcommands include:
			▪ add – Adds credentials to a user. For example:
			`credentials add password=<string> realm=<string> uid=<string>`
			▪ `addAll` – Adds credentials for all of the configured realms in the system to a user. For example:
			`credentials addAll password=<string> uid=<string>`
			▪ `delete` – Deletes realm credentials for a user. For example:
			`credentials delete realm=<string> uid=<string>`
			▪ `deleteAll` – Deletes all credentials for a user. For example:
			`credentials deleteall uid=<string>`
			▪ `update` – Updates the credentials for a user. For example:
			`credentials update password=<string> realm=<string> uid=<string>`
			▪ `updateAll` – Updates a user's credentials for all provisioned realms in the system. For example:
			`credentials updateAll password=<string> uid=<string>`
			▪ `list` – Lists all of the realms for which credentials exist for a given user. For example:
			`credentials list uid=<string>`

`identity add`	Enables you to create a basic user account.	None	None. See "Creating a User with the Identity Add Command".

7.3.1.1 Viewing Subcommands

To view the subcommands for a specific command, enter `help <command>`. For example, entering *`help`* for the `account` command (`help account`) retrieves a brief overview of the subcommands available to the `account` command (illustrated in Example 7–2).

Example 7–2 Retrieving Help for a Specific Command

```
*** Description ****
Contains commands for management of user accounts.
In an account you can set if the account is active,
```

```
locked or if it perhaps should be a temporarily account.

Aliases: [no aliases]

Syntax:
account

Sub-commands:
# Adds a new account to the system
  account add uid=<string> [ active=<true|false> ] [ locked=<true|false> ] [
accountExpiresAt=<accountExpiresAt> ] [ tempAccount=<true|false> ] [
description=<string> ] [ lockExpiresAt=<lockExpiresAt> ] [
currentFailedLogins=<integer> ]

# Deletes an account
  account delete uid=<string>

# Updates an account
  account update uid=<string> [ active=<true|false> ] [ locked=<true|false> ] [
accountExpiresAt=<accountExpiresAt> ] [ tempAccount=<true|false> ] [
description=<string> ] [ lockExpiresAt=<lockExpiresAt> ] [
currentFailedLogins=<integer> ]

# Retrieve information about a particular account
  account info uid=<string>
```

In addition to the overview of the command group, the information displayed by entering `help <command>` also includes the aliases (if any) to the command. For example, the overview of the account command illustrated in Example 7–2 notes `[no aliases]` for the command.

Note: The `delete` command used with `account`, `role`, `role system`, `role user`, `privateIdentity`, `publicIdentity`, and `identity` has the following aliases:

- `remove`
- `del`
- `rm`

Some commands require parameters. For example, if you enter `help role system add`, the system informs you that the `add` command requires the name of the role and an optional command for setting the description as well by displaying

```
role system add name=<string> [description=<string>].
```

Note: Optional commands such as `[description=<string>]` are enclosed within square brackets `[...]`.

The system alerts you if you omit a mandatory parameter or if you pass in a parameter that is not recognized.

7.4 Creating a User

This section describes the `publicIdentity` and `privateIdentity` commands and how to use them in conjunction with the `add`, `account`, `role`, and `credentials` subcommands listed in Table 7–1 to provision a user account to the Oracle database.

The Private Identity (`privateIdentity`) uniquely identifies a user within a given authentication realm. The Public Identity (`publicIdentity`) is the SIP address that users enter to register devices. This address is the user's AOR (Address of Record), and the means through which users call one another. A user can have only one Private Identity, but can have several Public Identities associated with that Private Identity.

> **Note:** To enable authentication to third-party databases (such as RADIUS), user accounts that contain authentication data and are stored externally must match the Private Identity to ensure the proper functioning of the Proxy Registrar and other applications that require authentication.

To create a user, first add the user to the system by creating a private identity and then a public identity for the user using the `privateIdentity` and `publicIdentity` commands with the `add privateId` and `add publicId` subcommands, respectively.

Once you create the private and public identity for the user, create an account for the user with the `account add uid` command and optionally set the status of the account (such as active or locked). The `role` command sets the role memberships for role-based permissions. Set the level of permissions for the users using the `role` command, and then set user credentials by defining the user's realm and password with the `credentials` command.

7.4.1 Creating a User from the Sash Command-Line Prompt

This section illustrates how to create a user from the Sash command prompt (`sash#`, illustrated in Example 7–3) by creating an OWLCS user known as *alice* using the commands described in Table 7–1.

1. Create a user using the `privateIdentity` command as follows:

   ```
   privateIdentity add privateId=alice
   ```

2. Create the public identity for alice by entering the SIP address:

   ```
   publicIdentity add publicId=sip:alice@test.company.com privateId=alice
   ```

3. Add an account for alice and use one of the optional commands described in Table 7–1 to set the status of the account. To create an active account for alice, enter the following:

   ```
   account add uid=alice active=true
   ```

Note: OWLCS Version 10.1.3.2 requires that the `uid` be in lower-case. Oracle Communicator users provisioned using OWLCS Version 10.1.3.2 must also enter their account names in lower case during login. OWLCS Version 10.1.3.3 and 10.1.3.4 support mixed-case `uids`. However, Oracle Communicator users can only log in by entering their user name exactly as it was provisioned. For example, if you define the `uid` as Alice, then the user must login as Alice. If you upgrade to 10.1.3.4 from 10.1.3.2, users provisioned in 10.1.3.2 must continue to log in using lower case.

4. Use the `role` command to add alice to the *Location Service* user group. Doing so grants alice permission to the Proxy Registrar's Location Service lookup:

   ```
   role user add uid=alice name="Location Service"
   ```

5. Add user authentication credentials for alice:

   ```
   credentials add uid=alice realm=test.company.com password=welcome1
   ```

   The `credentials` command is not needed for applications configured to use the RADIUS Login Module to authenticate users against RADIUS servers. Fore more information on these login modules, see Chapter 5, "Administering Security Features".

Note: You must also configure `realms` using the SIP Servlet Container MBean before you use Sash to add authorization credentials to a user.

Example 7–3 Creating a User from the Sash Command-Line Prompt

```
sash# privateIdentity add privateId=alice
sash# publicIdentity add publicId=sip:alice@test.company.com privateId=alice
sash# account add uid=alice active=true
sash# role user add uid=alice name="Location Service"
sash# credentials add uid=alice realm=test.company.com password=welcome1
```

Tip: You can create multiple users by creating Sash batch files. For more information, see "Scripting with Sash".

7.4.2 Creating a User with the Command Service MBean

You can execute Sash commands using the CommandService MBean's *execute* operation. The Command Service MBean is defined within the `subscrdataservcommandsear` application.

To create a user:

1. Select the *execute* operation. The *Operation* page for the *execute* operation appears.
2. Enter *privateIdentity add privateId=alice* in the *Value* field.
3. Click **Invoke Operation**. Repeat this process for each of the user creation commands. For example, the subsequent `publicIdentity` and `account` commands would both be followed by **Invoke Operation**.

7.4.3 Creating a User with the Identity Add Command

The `identity add` command enables you to create a user with one command string. This command, which is an alias to the `privateIdentity`, `publicIdentity`, `account`, `role` and `credentials` commands, enables you to quickly create a basic user account that contains the minimum information needed for users to connect to OWLCS through a SIP client. For example, to create a basic account for user *alice* using this command, enter the following from either the command line or through the Command Service Mbean's *execute* operation:

```
identity add privateId=alice publicId=sip:sip.alice@company.com role="Location
Service" realm=company.com password=welcome1
```

Note: For applications configured to authenticate users against a RADIUS system (the applications with the RADIUS Login Module as the security provider), the command to create a user account is as follows:

```
identity add privateId=alice publicId=sip:sip.alice@company.com
role="Location Service"
```

The `identity add` command only enables you to create a basic user account. Accounts that require more complex construction, such as those that associate multiple `publicId`s with a single `privateId`, must be created using multiple Sash commands as illustrated in Example 7–3.

7.4.3.1 Deleting a User Account with the identity delete Command

The `identity delete` command enables you to delete all of a user's roles, credentials, account information, public and private identities using a single command string. For example, to delete an account for user *alice* using this command, enter the following from either the command line or through the Command Service Mbean's *execute* operation:

```
identity delete privateId=alice
```

7.5 Provisioning the XDMS Using Sash

The commands for provisioning the XDMS are included in the `xcap` group. Each of these commands is preceded by `xcap`. The XDMS commands within the `xcap` group that support user provisioning are included in the `user` and `applicationUsage` subgroups. You can provision XDMS from the Sash prompt or by using the CommandService MBean that is provided with the Presence application.

7.5.1 Provisioning XDMS User Accounts Using the CommandService MBean

You can provision XDMS using the *execute* command provided by the CommandService MBean that is registered to the Presence application. Use the CommandService MBean's execute operation as described in "Creating a User with the Command Service MBean" to provision accounts to the XDMS. For more information on the MBean itself, see "Command Service (XDMS Provisioning)".

7.5.2 Provisioning XDMS User Accounts from the Sash Prompt

To use XDMS commands to provision users and application usages from the Sash prompt, you must first connect to an application that consumes XDMS, such as Presence.

For Windows systems, enter the following from the command prompt:

```
launch_sash.bat -a <application name>
```

On Linux, enter the following:

```
launch_sash.sh -a <application name>
```

For example, to connect to the Presence application on a Windows system:

1. From the command prompt, navigate to the `sbin` directory that contains the Sash executable. This file is located at *MIDDLEWARE_HOME/as11gr1wlcs/communications/sash/sbin*.
2. Enter the name of the Presence application using `sash.bat -a presenceapplication`.
3. When prompted, login to Sash using your OWLCS administrator name and password.
4. From the Sash command prompt, enter an XDMS command, such as `xcap user list`.

7.5.3 Using xcap Commands

This section describes how to manage user accounts and application usages using the `xcap` group of commands.

7.5.3.1 Provisioning XDMS User Accounts

The `add`, `delete` and `list` commands enable you to manage user accounts on the XDMS.

7.5.3.2 Adding XDMS Users

The `xcap user add` command adds an XDMS user with the given user name and application usage. For example, to add a user from the Sash prompt, enter:

```
sash# xcap user add userName=<string> appusage=<string>
```

> **Note:** Do not use the `add` command if the XDMS is configured to automatically create users. For more information on configuring XCAP, see Section 9.2.9, "XCapConfigManager".

7.5.3.3 Removing an XDMS User

The `xcap user delete` command removes an XDMS user with the given user name (and all existing documents) from the application usage. For example, to delete a user from the Sash prompt, enter:

```
sash# xcap user delete userName=<string> [ appusages=<string> ]
```

The application usage parameter (`appusages`) is optional. If no application usage is specified, then the user is removed from all application usages.

7.5.3.4 Searching for Application Usage for an XDMS User

The `xcap user appusages` command returns all the application usages applicable to a given user. To review the application usages assigned to a user, enter the following from the Sash prompt:

```
sash# xcap user appusages userName=<string>
```

7.5.3.5 Listing XDMS Users

The `xcap user list` command returns all of the XDMS users in the system, or optionally returns the XDMS users for a given application usage.

```
sash# xcap user list [ all=<true|false> ] [ appusage=<string> ]
```

If the optional `all` parameter is not set, then the resulting display is limited to a maximum of 100 users.

7.5.3.6 Provisioning Application Usage

The commands for provisioning of XDMS application usage are in the `appusage` group (`xcap appusage`). Three types of applications are supported: `resource-lists_au.xml`, `pidfmanipulation_au.xml` and `presrules_au.xml`.

The SASH command for adding application usage is:

```
xcap appusage create appusage=<application_usage>
configurationFilename=<application>_au.xml
```

Three types of applications are supported as shown in.

Table 7–2 Supported Application Types

applicationUsage	configurationFilename
resource-lists	`resource-lists_au.xml`
pidf-manipulation	`pidfmanipulation_au.xml`
pres-rules	`presrules_au.xml`

7.5.3.7 Listing All Application Usages

The `xcap appusage list` command returns all the application usages on the server. For example, enter the following from the Sash prompt:

```
sash# xcap appusage list
```

7.6 Scripting with Sash

You can construct scripts for common tasks that contain several operations. Sash can be evoked to execute a file containing a list of commands. To enable scripting, Sash provides such command-line flags as:

- `-- exec` (short name: `-e`): When this command-line flag is followed by a command enclosed within quotation marks, Sash executes the command and then exits.
- `-- file` (short name: `-f`): When this command-line flag is followed by a filename, Sash reads the file and executes all commands in the file as they were entered and then exits.

Example 7–4 illustrates a text file called ocsm_users.txt, which contains a group of users defined with the `identity add` command. You can provision these users by entering `-f OWLCS_users.txt` from the Sash prompt:

Example 7–4 Creating Users from a Text File (OWLCS_users.txt)

```
identity add privateId=candace publicId=sip:candace@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=deirdre publicId=sip:deirdre@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=evelyn publicId=sip:evelyn@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=frank publicId=sip:frank@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
```

- `-- nonewline`: This command-line flag facilitates parsing output by stripping returns or newlines from the messages returned from the executed commands. Although this command facilitates parsing, it makes reading messages manually more difficult.

7.7 Error Logging in Sash

Sash does not log to any files (with the default configuration), it only prints messages on the console. The log level for Sash is configured in $ORACLE _ HOME/as11gr1wlcs1/communications/sash/conf.

8 Monitoring and Troubleshooting

The following sections describe how to configure use the Oracle WebLogic Communication Services "echo server" process to improve SIP data tier failover performance when a server becomes physically disconnected from the network:

8.1 Avoiding and Recovering from Server Failures

A variety of events can lead to the failure of a server instance. Often one failure condition leads to another. Loss of power, hardware malfunction, operating system crashes, network partitions, or unexpected application behavior may each contribute to the failure of a server instance.

Oracle WebLogic Communication Services uses a highly clustered architecture as the basis for minimizing the impact of failure events. However, even in a clustered environment it is important to prepare for a sound recovery process in the event that an individual server or server machine fails.

The following sections summarize Oracle WebLogic Communication Services failure prevention and recovery features, and describe the configuration artifacts that are required in order to restore different portions of a Oracle WebLogic Communication Services domain

8.1.1 Failure Prevention and Automatic Recovery Features

Oracle WebLogic Communication Services, and the underlying WebLogic Server platform, provide many features that protect against server failures. In a production system, all available features should be used in order to ensure uninterrupted service.

8.1.1.1 Overload Protection

Oracle WebLogic Communication Services detects increases in system load that could affect the performance and stability of deployed SIP Servlets, and automatically throttles message processing at predefined load thresholds.

Using overload protection helps you avoid failures that could result from unanticipated levels of application traffic or resource utilization.

Oracle WebLogic Communication Services attempts to avoid failure when certain conditions occur:

- The rate at which SIP sessions are created reaches a configured value, or
- The size of the SIP timer and SIP request-processing execute queues reaches a configured length.

The underlying WebLogic Server platform also detects increases in system load that can affect deployed application performance and stability. WebLogic Server allows administrators to configure failure prevention actions that occur automatically at predefined load thresholds. Automatic overload protection helps you avoid failures that result from unanticipated levels of application traffic or resource utilization as indicated by:

- A workload manager's capacity being exceeded
- The HTTP session count increasing to a predefined threshold value
- Impending out of memory conditions

8.1.1.2 Redundancy and Failover for Clustered Services

You can increase the reliability and availability of your applications by using multiple engine tier servers in a dedicated cluster, as well as multiple SIP data tier servers (replicas) in a dedicated SIP data tier cluster. Because engine tier clusters maintain no stateful information about SIP dialogs (calls), the failure of an engine tier server does not result in any data loss or dropped calls. Multiple replicas in a SIP data tier partition store redundant copies of call state information, and automatically failover to one another should a replica fail.

8.1.1.3 Automatic Restart for Failed Server Instances

WebLogic Server self-health monitoring features improve the reliability and availability of server instances in a domain. Selected subsystems within each server instance monitor their health status based on criteria specific to the subsystem. (For example, the JMS subsystem monitors the condition of the JMS thread pool while the core server subsystem monitors default and user-defined execute queue statistics.) If an individual subsystem determines that it can no longer operate in a consistent and reliable manner, it registers its health state as "failed" with the host server.

Each WebLogic Server instance, in turn, checks the health state of its registered subsystems to determine its overall viability. If one or more of its critical subsystems have reached the FAILED state, the server instance marks its own health state FAILED to indicate that it cannot reliably host an application.

When used in combination with Node Manager, server self-health monitoring enables you to automatically reboot servers that have failed. This improves the overall reliability of a domain, and requires no direct intervention from an administrator.

8.1.1.4 Managed Server Independence Mode

Managed Servers maintain a local copy of the domain configuration. When a Managed Server starts, it contacts its Administration Server to retrieve any changes to the domain configuration that were made since the Managed Server was last shut down. If a Managed Server cannot connect to the Administration Server during startup, it can use its locally-cached configuration information—this is the configuration that was current at the time of the Managed Server's most recent shutdown. A Managed Server that starts up without contacting its Administration Server to check for configuration updates is running in *Managed Server Independence (MSI)* mode. By default, MSI mode is enabled.

8.1.1.5 Automatic Migration of Failed Managed Servers

When using Linux or UNIX operating systems, you can use WebLogic Server's server migration feature to automatically start a candidate (backup) server if a Network tier server's machine fails or becomes partitioned from the network. The server migration feature uses node manager, in conjunction with the `wlsifconfig.sh` script, to automatically boot candidate servers using a floating IP address. Candidate servers are booted only if the primary server hosting a Network tier instance becomes unreachable. See "Whole Server Migration" in *Oracle Fusion Middleware Using Clusters for Oracle WebLogic Server* documentation for more information about using the server migration feature.

8.1.1.6 Geographic Redundancy for Regional Site Failures

In addition to server-level redundancy and failover capabilities, Oracle WebLogic Communication Services enables you to configure peer sites to protect against catastrophic failures, such as power outages, that can affect an entire domain. This enables you to failover from one geographical site to another, avoiding complete service outages.

8.1.2 Directory and File Backups for Failure Recovery

Recovery from the failure of a server instance requires access to the domain's configuration data. By default, the Administration Server stores a domain's primary configuration data in a file called `domain_name/config/config.xml`, where `domain_name` is the root directory of the domain. The primary configuration file may reference additional configuration files for specific WebLogic Server services, such as JDBC and JMS, and for Oracle WebLogic Communication Services services, such as SIP container properties and SIP data tier configuration. The configuration for specific services are stored in additional XML files in subdirectories of the `domain_name/config` directory, such as `domain_name/config/jms`, `domain_name/config/jdbc`, and `domain_name/config/custom` for Oracle WebLogic Communication Services configuration files.

The Administration Server can automatically archive multiple versions of the domain configuration (the entire `domain-name/config` directory). The configuration archives can be used for system restoration in cases where accidental configuration changes need to be reversed. For example, if an administrator accidentally removes a configured resource, the prior configuration can be restored by using the last automated backup.

The Administration Server stores only a finite number of automated backups locally in `domain_name/config`. For this reason, automated domain backups are limited in their ability to guard against data corruption, such as a failed hard disk. Automated backups also do not preserve certain configuration data that are required for full domain restoration, such as LDAP repository data and server start-up scripts. Oracle

recommends that you also maintain multiple backup copies of the configuration and security offline, in a source control system.

This section describes file backups that Oracle WebLogic Communication Services performs automatically, as well as manual backup procedures that an administrator should perform periodically.

8.1.2.1 Enabling Automatic Configuration Backups

Follow these steps to enable automatic domain configuration backups on the Administration Server for your domain:

1. Access the Administration Console for your domain.
2. In the left pane of the Administration Console, select the name of the domain.
3. In the right pane, click the **Configuration > General** tab.
4. Select **Advanced** to display advanced options.
5. Select **Configuration Archive Enabled**.
6. In the Archive Configuration Count box, enter the maximum number of configuration file revisions to save.
7. Click **Save**.

When you enable configuration archiving, the Administration Server automatically creates a configuration JAR file archive. The JAR file contains a complete copy of the previous configuration (the complete contents of the `domain-name\config` directory). JAR file archive files are stored in the `domain-name\configArchive` directory. The files use the naming convention `config-number.jar`, where `number` is the sequential number of the archive.

When you save a change to a domain's configuration, the Administration Server saves the previous configuration in `domain-name\configArchive\config.xml#n`. Each time the Administration Server saves a file in the `configArchive` directory, it increments the value of the `#n` suffix, up to a configurable number of copies—5 by default. Thereafter, each time you change the domain configuration:

- The archived files are rotated so that the newest file has a suffix with the highest number,
- The previous archived files are renamed with a lower number, and
- The oldest file is deleted.

Keep in mind that configuration archives are stored locally within the domain directory, and they may be overwritten according to the maximum number of revisions you selected. For these reasons, you must also create your own off-line archives of the domain configuration, as described in Section 8.1.2.2, "Storing the Domain Configuration Offline".

8.1.2.2 Storing the Domain Configuration Offline

Although automatic backups protect against accidental configuration changes, they do not protect against data loss caused by a failure of the hard disk that stores the domain configuration, or accidental deletion of the domain directory. To protect against these failures, you must also store a complete copy of the domain configuration offline, preferably in a source control system.

Oracle recommends storing a copy of the domain configuration at regular intervals. For example, back up a new revision of the configuration when:

- you first deploy the production system
- you add or remove deployed applications
- the configuration is tuned for performance
- any other permanent change is made.

The domain configuration backup should contain the complete contents of the `domain_name/config` directory. For example:

```
cd ~/user_projects/domains/mydomain
tar cvf domain-backup-06-17-2007.jar config
```

Store the new archive in a source control system, preserving earlier versions should you need to restore the domain configuration to an earlier point in time.

8.1.2.3 Backing Up Server Start Scripts

In a Oracle WebLogic Communication Services deployment, the start scripts used to boot engine and SIP data tier servers are generally customized to include domain-specific configuration information such as:

- JVM Garbage Collection parameters required to achieve throughput targets for SIP message processing (see Section 8.8, "Tuning JVM Garbage Collection for Production Deployments"). Different parameters (and therefore, different start scripts) are generally used to boot engine and SIP data tier servers.
- Configuration parameters and startup information for the Oracle WebLogic Communication Services heartbeat mechanism. If you use the heartbeat mechanism, engine tier server start scripts should include startup options to enable and configure the heartbeat mechanism. SIP data tier server start scripts should include startup options to enable heartbeats and start the `WlssEchoServer` process.

Backup each distinct start script used to boot engine tier, SIP data tier, or diameter relay servers in your domain.

8.1.2.4 Backing Up Logging Servlet Applications

If you use Oracle WebLogic Communication Services logging Servlets (see Section 8.7, "Logging SIP Requests and Responses") to perform regular logging or auditing of SIP messages, backup the complete application source files so that you can easily redeploy the applications should the staging server fail or the original deployment directory becomes corrupted.

8.1.2.5 Backing Up Security Data

The WebLogic Security service stores its configuration data `config.xml` file, and also in an LDAP repository and other files.

8.1.2.5.1 Backing Up SerializedSystemIni.dat and Security Certificates All servers create a file named `SerializedSystemIni.dat` and place it in the server's root directory. This file contains encrypted security data that must be present to boot the server. You must back up this file.

If you configured a server to use SSL, also back up the security certificates and keys. The location of these files is user-configurable.

8.1.2.5.2 Backing Up the WebLogic LDAP Repository The default Authentication, Authorization, Role Mapper, and Credential Mapper providers that are installed with

Oracle WebLogic Communication Services store their data in an LDAP server. Each Oracle WebLogic Communication Services contains an embedded LDAP server. The Administration Server contains the master LDAP server, which is replicated on all Managed Servers. If any of your security realms use these installed providers, you should maintain an up-to-date backup of the following directory tree:

```
domain_name\adminServer\ldap
```

where `domain_name` is the domain's root directory and `adminServer` is the directory in which the Administration Server stores runtime and security data.

Each Oracle WebLogic Communication Services has an LDAP directory, but you only need to back up the LDAP data on the Administration Server—the master LDAP server replicates the LDAP data from each Managed Server when updates to security data are made. WebLogic security providers cannot modify security data while the domain's Administration Server is unavailable. The LDAP repositories on Managed Servers are replicas and cannot be modified.

The `ldap/ldapfiles` subdirectory contains the data files for the LDAP server. The files in this directory contain user, group, group membership, policies, and role information. Other subdirectories under the `ldap` directory contain LDAP server message logs and data about replicated LDAP servers.

Do not update the configuration of a security provider while a backup of LDAP data is in progress. If a change is made—for instance, if an administrator adds a user—while you are backing up the `ldap` directory tree, the backups in the `ldapfiles` subdirectory could become inconsistent. If this does occur, consistent, but potentially out-of-date, LDAP backups are available.

Once a day, a server suspends write operations and creates its own backup of the LDAP data. It archives this backup in a `ZIP` file below the `ldap\backup` directory and then resumes write operations. This backup is guaranteed to be consistent, but it might not contain the latest security data.

8.1.2.6 Backing Up Additional Operating System Configuration Files

Certain files maintained at the operating system level are also critical in helping you recover from system failures. Consider backing up the following information as necessary for your system:

- Load Balancer configuration scripts. For example, any automated scripts used to configure load balancer pools and virtual IP addresses for the engine tier cluster, as well as NAT configuration settings.
- NTP client configuration scripts used to synchronize the system clocks of engine and SIP data tier servers.
- Host configuration files for each Oracle WebLogic Communication Services machine (host names, virtual and real IP addresses for multi-homed machines, IP routing table information).

8.1.3 Restarting a Failed Administration Server

When you restart a failed Administration Server, no special steps are required. Start the Administration Server as you normally would.

If the Administration Server shuts down while Managed Servers continue to run, you do not need to restart the Managed Servers that are already running in order to recover management of the domain. The procedure for recovering management of an

active domain depends upon whether you can restart the Administration Server on the same machine it was running on when the domain was started.

8.1.3.1 Restarting an Administration Server on the Same Machine

If you restart the WebLogic Administration Server while Managed Servers continue to run, by default the Administration Server can discover the presence of the running Managed Servers.

> **Note:** Make sure that the startup command or startup script does not include `-Dweblogic.management.discover=false`, which disables an Administration Server from discovering its running Managed Servers.

The root directory for the domain contains a file, `running-managed-servers.xml`, which contains a list of the Managed Servers in the domain and describes whether they are running or not. When the Administration Server restarts, it checks this file to determine which Managed Servers were under its control before it stopped running.

When a Managed Server is gracefully or forcefully shut down, its status in `running-managed-servers.xml` is updated to "not-running". When an Administration Server restarts, it does not try to discover Managed Servers with the "not-running" status. A Managed Server that stops running because of a system crash, or that was stopped by killing the JVM or the command prompt (shell) in which it was running, will still have the status "running' in `running-managed-servers.xml`. The Administration Server will attempt to discover them, and will throw an exception when it determines that the Managed Server is no longer running.

Restarting the Administration Server does not cause Managed Servers to update the configuration of static attributes. *Static attributes* are those that a server refers to only during its startup process. Servers instances must be restarted to take account of changes to static configuration attributes. Discovery of the Managed Servers only enables the Administration Server to monitor the Managed Servers or make runtime changes in attributes that can be configured while a server is running (dynamic attributes).

8.1.3.2 Restarting an Administration Server on Another Machine

If a machine crash prevents you from restarting the Administration Server on the same machine, you can recover management of the running Managed Servers as follows:

1. Install the Oracle WebLogic Communication Services software on the new administration machine (if this has not already been done).
2. Make your application files available to the new Administration Server by copying them from backups or by using a shared disk. Your application files should be available in the same relative location on the new file system as on the file system of the original Administration Server.
3. Make your configuration and security data available to the new administration machine by copying them from backups or by using a shared disk. For more information, refer to Section 8.1.2.2, "Storing the Domain Configuration Offline" and Section 8.1.2.5, "Backing Up Security Data".
4. Restart the Administration Server on the new machine.

Make sure that the startup command or startup script does not include `-Dweblogic.management.discover=false`, which disables an Administration Server from discovering its running Managed Servers.

When the Administration Server starts, it communicates with the Managed Servers and informs them that the Administration Server is now running on a different IP address.

8.1.4 Restarting Failed Managed Servers

If the machine on which the failed Managed Server runs can contact the Administration Server for the domain, simply restart the Managed Server manually or automatically using Node Manager. Note that you must configure Node Manager and the Managed Server to support automated restarts.

If the Managed Server cannot connect to the Administration Server during startup, it can retrieve its configuration by reading locally-cached configuration data. A Managed Server that starts in this way is running in Managed Server Independence (MSI) mode.

To start up a Managed Server in MSI mode:

1. Ensure that the following files are available in the Managed Server's root directory:
   - `msi-config.xml`
   - `SerializedSystemIni.dat`
   - `boot.properties`

   If these files are not in the Managed Server's root directory:
   a. Copy the `config.xml` and `SerializedSystemIni.dat` file from the Administration Server's root directory (or from a backup) to the Managed Server's root directory.
   b. Rename the configuration file to `msi-config.xml`. When you start the server, it will use the copied configuration files.

   Note: Alternatively, use the `-Dweblogic.RootDirectory=path` startup option to specify a root directory that already contains these files.

   ---

2. Start the Managed Server at the command line or using a script.

   The Managed Server will run in MSI mode until it is contacted by its Administration Server. For information about restarting the Administration Server in this scenario, see Section 8.1.3, "Restarting a Failed Administration Server".

8.2 Overview of Failover Detection

In a production system, engine tier servers continually access SIP data tier replicas in order to retrieve and write call state data. The Oracle WebLogic Communication Services architecture depends on engine tier nodes to detect when a SIP data tier server has failed or become disconnected. When an engine cannot access or write call state data because a replica is unavailable, the engine connects to another replica in the same partition and reports the offline server. The replica updates the current view of

the SIP data tier to account for the offline server, and other engines are then notified of the updated view as they access and retrieve call state data.

By default, an engine tier server uses its RMI connection to the replica to determine if the replica has failed or become disconnected. The algorithms used to determine a failure of an RMI connection are reliable, but ultimately they depend on the TCP protocol's retransmission timers to diagnose a disconnection (for example, if the network cable to the replica is removed). Because the TCP retransmission timer generally lasts a full minute or longer, Oracle WebLogic Communication Services provides an alternate method of detecting failures that can diagnose a disconnected replica in a matter of a few seconds.

8.2.1 WlssEchoServer Failure Detection

`WlssEchoServer` is a separate process that you can run on the same server hardware as a SIP data tier replica. The purpose of `WlssEchoServer` is to provide a simple UDP echo service to engine tier nodes to be used for determining when a SIP data tier server goes offline, for example in the event that the network cable is disconnected. The algorithm for detecting failures with `WlssEchoServer` is as follows:

1. For all normal traffic, engine tier servers communicate with SIP data tier replicas using TCP. TCP is used as the basic transport between the engine tier and SIP data tier regardless of whether or not `WlssEchoServer` is used.
2. Engine tier servers send a periodic heartbeat message to each configured `WlssEchoServer` over UDP. During normal operation, `WlssEchoServer` responds to the heartbeats so that the connection between the engine node and replica is verified.
3. Should there be a complete failure of the SIP data tier stack, or the network cable is disconnected, the heartbeat messages are not returned to the engine node. In this case, the engine node can mark the replica as being offline *without* having to wait for the normal TCP connection timeout.
4. After identifying the offline server, the engine node reports the failure to an available SIP data tier replica, and the SIP data tier view is updated as described in the previous section.

Also, should a SIP data tier server notice that its local `WlssEchoServer` process has died, it automatically shuts down. This behavior ensures even quicker failover because avoids the time it takes engine nodes to notice and report the failure as described in Section 8.2, "Overview of Failover Detection".

You can configure the heartbeat mechanism on engine tier servers to increase the performance of failover detection as necessary. You can also configure the listen port and log file that `WlssEchoServer` uses on SIP data tier servers.

8.2.2 Forced Shutdown for Failed Replicas

If any engine tier server cannot communicate with a particular replica, the engine access another, available replica in the SIP data tier to report the offline server. The replica updates its view of the affected partition to remove the offline server. The updated view is then distributed to all engine tier servers that later access the partition. Propagating the view in this manner helps to ensure that engine servers do not attempt to access the offline replica.

The replica that updates the view also issues a one-time request to the offline replica to ask it to shut down. This is done to try to shut-down running replica servers that

cannot be accessed by one or more engine servers due to a network outage. If an active replica can reach the replica marked as "offline," the offline replica shuts down.

8.3 Improving Failover Performance for Physical Network Failures

> **Note:** Using `WlssEchoServer` is not required in all Oracle WebLogic Communication Services installations. Enable the echo server only when your system requires detection of a network or replica failure faster than the configured TCP timeout interval.

Observe the following requirements and restrictions when using `WlssEchoServer` to detect replica failures:

- If you use the heartbeat mechanism to detect failures, you must ensure that the `WlssEchoServer` process is always running on each replica server machine. If the `WlssEchoServer` process fails or is stopped, the replica will be treated as being "offline" even if the server process is unaffected.
- Note that `WlssEchoServer` listens on all IP addresses available on the server machine.
- `WlssEchoServer` requires a dedicated port number to listen for heartbeat messages.

8.3.1 Starting WlssEchoServer on SIP Data Tier Server Machines

`WlssEchoServer` is a Java program that you can start directly from a shell or command prompt. The basic syntax for starting `WlssEchoServer` is:

```
java -classpath WLSS_HOME/server/lib/wlss/wlssechosvr.jar options
com.bea.wcp.util.WlssEchoServer
```

Where `WLSS_HOME` is the path to the Oracle WebLogic Communication Services installation and `options` may include one of the options described in Table 8–1.

Table 8–1 WlssEchoServer Options

Option	Description
`-Dwlss.ha.echoserver.ipaddress`	Specifies the IP address on which the `WlssEchoServer` instance listens for heartbeat messages. If you do not specify an IP address, the instance listens on any available IP address (0.0.0.0).
`-Dwlss.ha.echoserver.port`	Specifies the port number used to listen for heartbeat messages. Ensure that the port number you specify is not used by any other process on the server machine. By default `WlssEchoServer` uses port 6734.
`-Dwlss.ha.echoserver.logfile`	Specifies the log file location and name. By default, log messages are written to `./echo_servertime.log` where `time` is the time expressed in milliseconds.

Oracle recommends that you include the command to start `WlssEchoServer` in the same script you use to start each Oracle WebLogic Communication Services SIP data tier instance. If you use the `startManagedWebLogic.sh` script to start an engine or SIP data tier server instance, add a command to start `WlssEchoServer` before the final command used to start the server. For example, change the lines:

```
"$JAVA_HOME/bin/java" ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}     \
  -Dweblogic.Name=${SERVER_NAME}                                  \
  -Dweblogic.management.username=${WLS_USER}                      \
  -Dweblogic.management.password=${WLS_PW}                        \
  -Dweblogic.management.server=${ADMIN_URL}                       \
  -Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy" \
   weblogic.Server
```

to read:

```
"$JAVA_HOME/bin/java" -classpath WLSS_HOME/server/lib/wlss/wlssechosvr.jar    \
  -Dwlss.ha.echoserver.ipaddress=192.168.1.4                    \
  -Dwlss.ha.echoserver.port=6734 com.bea.wcp.util.WlssEchoServer &
"$JAVA_HOME/bin/java" ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}     \
  -Dweblogic.Name=${SERVER_NAME}                                  \
  -Dweblogic.management.username=${WLS_USER}                      \
  -Dweblogic.management.password=${WLS_PW}                        \
  -Dweblogic.management.server=${ADMIN_URL}                       \
  -Djava.security.policy="${WL_HOME}/server/lib/weblogic.policy" \
   weblogic.Server
```

8.3.2 Enabling and Configuring the Heartbeat Mechanism on Servers

To enable the `WlssEchoServer` heartbeat mechanism, you must include the `-Dreplica.host.monitor.enabled` JVM argument in the command you use to start all engine and SIP data tier servers. Oracle recommends adding this option directly to the script used to start Managed Servers in your system. For example, in the `startManagedWebLogic.sh` script, change the line:

```
# JAVA_OPTIONS="-Dweblogic.attribute=value -Djava.attribute=value"
```

to read:

```
JAVA_OPTIONS="-Dreplica.host.monitor.enabled=true"
```

Several additional JVM options configure the functioning of the heartbeat mechanism. Table 8–1 describes the options used to configure failure detection.

Table 8–2 WlssEchoServer Options

Option	Description
`-Dreplica.host.monitor.enabled`	This system property is required on both engine and SIP data tier servers to enable the heartbeat mechanism.
`-Dwlss.ha.heartbeat.interval`	Specifies the number of milliseconds between heartbeat messages. By default heartbeats are sent every 1,000 milliseconds.
`-Dwlss.ha.heartbeat.count`	Specifies the number of consecutive, missed heartbeats that are permitted before a replica is determined to be offline. By default, a replica is marked offline if the `WlssEchoServer` process on the server fails to respond to 3 heartbeat messages.
`-Dwlss.ha.heartbeat.SoTimeout`	Specifies the UDP socket timeout value.

8.4 Configuring SNMP

Oracle WebLogic Communication Services includes a dedicated SNMP MIB to monitor activity on engine tier and SIP data tier server instances. The Oracle WebLogic Communication Services MIB is available on both Managed Servers and the

Administration Server of a domain. However, Oracle WebLogic Communication Services engine and SIP data tier traps are generated only by the Managed Server instances that make up each tier. If your Administration Server is not a target for the `sipserver` custom resource, it will generate only WebLogic Server SNMP traps (for example, when a server in a cluster fails). Administrators should monitor both WebLogic Server and Oracle WebLogic Communication Services traps to evaluate the behavior of the entire domain.

Note: Oracle WebLogic Communication Services MIB objects are read-only. You cannot modify a Oracle WebLogic Communication Services configuration using SNMP.

8.4.1 Browsing the MIB

The Oracle WebLogic Communication Services MIB file is installed in *`WLSS_HOME`*`/server/lib/wlss/BEA-WLSS-MIB.asn1`. Use an available SNMP management tool or MIB browser to view the contents of this file. See also Section 8.5.2, "Trap Descriptions" for a description of common SNMP traps.

8.4.2 Steps for Configuring SNMP

To enable SNMP monitoring for the entire Oracle WebLogic Communication Services domain, follow these steps:

1. Login to the Administration Console for the Oracle WebLogic Communication Services domain.
2. In the left pane, select the **Diagnostics > SNMP** node.
3. In the Server SNMP Agents table, click the **New** button to create a new agent.

Note: Ensure that you create a new Server SNMP agent, rather than a Domain-Scoped agent.

---

4. Enter a unique name for the new SNMP agent (for example, "engine1snmp") and click **OK**.
5. Select the newly-created SNMP agent from the Server SNMP Agents table.
6. On the **Configuration > General** tab:
   a. Select the Enabled check box to enable the agent.
   b. Enter an unused port number in the SNMP UDP Port field.

Note: If you run multiple Managed Server instances on the same machine, each server instance must use a dedicated SNMP agent with a unique SNMP port number.

---

   c. Click **Save**.
7. Repeat the above steps to generate a unique SNMP agent for each server in your deployment (SIP data tier server, engine tier server, and Administration Server).

8.5 Understanding and Responding to SNMP Traps

The following sections describe the Oracle WebLogic Communication Services SNMP traps in more detail. Recovery procedures for responding to individual traps are also included where applicable.

8.5.1 Files for Troubleshooting

The following Oracle WebLogic Communication Services log and configuration files are frequently helpful for troubleshooting problems, and may be required by your technical support contact:

- `$DOMAIN_DIR/config/config.xml`
- `$DOMAIN_DIR/config/custom/sipserver.xml`
- `$DOMAIN_DIR/servername/*.log (server and message logs)`
- `sip.xml (in the /WEB-INF subdirectory of the application)`
- `web.xml (in the /WEB-INF subdirectory of the application)`

General information that can help the technical support team includes:

- The specific versions of:
  - Oracle WebLogic Communication Services
  - Java SDK
  - Operating System
- Thread dumps for hung Oracle WebLogic Communication Services processes
- Network analyzer logs

8.5.2 Trap Descriptions

Table 8–3 lists the Oracle WebLogic Communication Services SNMP traps and indicates whether the trap is generated by servers in the engine tier or SIP data tier. Each trap is described in the sections that follow.

Table 8–3 Oracle WebLogic Communication Services SNMP Traps

Server Node in which Trap is Generated	Trap Name
Engine Tier Servers	Section 8.5.2.1, "connectionLostToPeer"
	Section 8.5.2.2, "connectionReestablishedToPeer"
	Section 8.5.2.4, "overloadControlActivated, overloadControlDeactivated"
	Section 8.5.2.9, "sipAppDeployed"
	Section 8.5.2.10, "sipAppUndeployed"
	Section 8.5.2.11, "sipAppFailedToDeploy"
Engine and SIP Data Tier Servers, if servers are members of a cluster	Section 8.5.2.8, "serverStopped"
SIP Data Tier Servers	Section 8.5.2.3, "dataTierServerStopped"

Table 8–3 (Cont.) Oracle WebLogic Communication Services SNMP Traps

Server Node in which Trap is Generated	Trap Name
	Section 8.5.2.5, "replicaAddedToPartition"
	Section 8.5.2.6, "replicaRemovedEnginesRegistration"
	Section 8.5.2.7, "replicaRemovedFromPartition"

8.5.2.1 connectionLostToPeer

This trap is generated by an engine tier server instance when it loses its connection to a replica in the SIP data tier. It may indicate a network connection problem between the engine and SIP data tiers, or may be generated with additional traps if a SIP data tier server fails.

8.5.2.1.1 Recovery Procedure If this trap occurs in isolation from other traps indicating a server failure, it generally indicates a network failure. Verify or repair the network connection between the affected engine tier server and the SIP data tier server.

If the trap is accompanied by additional traps indicating a SIP data tier server failure (for example, `dataTierServerStopped`), follow the recovery procedures for the associated traps.

8.5.2.2 connectionReestablishedToPeer

This trap is generated by an engine tier server instance when it successfully reconnects to a SIP data tier server after a prior failure (after a `connectionLostToPeer` trap was generated). Repeated instances of this trap may indicate an intermittent network failure between the engine and SIP data tiers.

8.5.2.2.1 Recovery Procedure See Section 8.5.2.1, "connectionLostToPeer".

8.5.2.3 dataTierServerStopped

Oracle WebLogic Communication Services SIP data tier nodes generate this alarm when an unrecoverable error occurs in a WebLogic Server instance that is part of the SIP data tier. Note that this trap may be generated by the server that is shutting down, by another replica in the same partition, or in some cases by both servers (network outages can sometimes trigger both servers to generate the same trap).

8.5.2.3.1 Recovery Procedure See the Recovery Procedure for Section 8.5.2.8, "serverStopped".

8.5.2.4 overloadControlActivated, overloadControlDeactivated

Oracle WebLogic Communication Services engine tier nodes use a configurable throttling mechanism that helps you control the number of new SIP requests that are processed. After a configured overload condition is observed, Oracle WebLogic Communication Services destroys new SIP requests by responding with "503 Service Unavailable" to the caller. The servers continues to destroy new requests until the overload condition is resolved according to a configured threshold control value. This alarm is generated when the throttling mechanism is activated. The throttling behavior should eventually return the server to a non-overloaded state, and further action may be unnecessary.

8.5.2.4.1 Recovery Procedure Follow this recovery procedure:

1. Check other servers to see if they are nearly overloaded.

2. Check to see if the load balancer is correctly balancing load across the application servers, or if it is overloading one or more servers. If additional servers are nearly overloaded, Notify Tier 4 support immediately.
3. If the issue is limited to one server, notify Tier 4 support within one hour.

8.5.2.4.2 Additional Overload Information If you set the queue length as an incoming call overload control, you can monitor the length of the queue using the Administration Console. If you specify a session rate control, you cannot monitor the session rate using the Administration Console. (The Administration Console only displays the current number of SIP sessions, not the rate of new sessions generated.)

8.5.2.5 replicaAddedToPartition

Oracle WebLogic Communication Services SIP data tier nodes generate this alarm when a server instance is added to a partition in the SIP data tier.

8.5.2.5.1 Recovery Procedure This trap is generated during normal startup procedures when SIP data tier servers are booted.

8.5.2.6 replicaRemovedEnginesRegistration

SIP data tier nodes generate this alarm if an engine server client that was not registered (or was removed from the list of registered engines) attempts to communicate with the SIP data tier. This trap is generally followed by a `serverStopped` trap indicating that the engine tier server was shut down to preserve SIP data tier consistency.

8.5.2.6.1 Recovery Procedure Restart the engine tier server. Repeated occurrences of this trap may indicate a network problem between the engine tier server and one or more replicas.

8.5.2.7 replicaRemovedFromPartition

Oracle WebLogic Communication Services SIP data tier nodes generate this alarm when a server is removed from the SIP data tier, either as a result of a normal shutdown operation or because of a failure. There must be at least one replica remaining in a partition to generate this trap; if a partition has only a single replica and that replica fails, the trap cannot be generated. In addition, because engine tier nodes determine when a replica has failed, an engine tier node must be running in order for this trap to be generated.

8.5.2.7.1 Recovery Procedure If this trap is generated as a result of a server instance failure, additional traps will be generated to indicate the exception. See the recovery procedures for traps generated in addition to `replicaRemovedFromPartition`.

8.5.2.8 serverStopped

This trap indicates that the WebLogic Server instance is now down. This trap applies to both engine tier and SIP data tier server instances, but only when the servers are members of a named WebLogic Server cluster. If this trap is received spontaneously and not as a result of a controlled shutdown, follow the steps below.

8.5.2.8.1 Recovery Procedure Follow this recovery procedure:

1. Use the following command to identify the hung process:

```
ps -ef | grep java
```

There should be only one PID for each WebLogic Server instance running on the machine.

2. After identifying the affected PID, use the following command to kill the process:

```
kill -3 [pid]
```

3. This command generates the actual thread dump. If the process is not immediately killed, repeat the command several times, spaced 5-10 seconds apart, to help diagnose potential deadlock problems, until the process is killed.
4. Attempt to restart Oracle WebLogic Communication Services immediately.
5. Make a backup copy of all SIP logs on the affected server to aid in troubleshooting. The location of the logs varies based on the server configuration.
6. Copy each log to assist Tier 4 support with troubleshooting the problem.

> **Note:** Oracle WebLogic Communication Services logs are truncated according to your system configuration. Make backup logs immediately to avoid losing critical troubleshooting information.

7. Notify Tier 4 support and include the log files with the trouble ticket.
8. Monitor the server closely over next 24 hours. If the source of the problem cannot be identified in the log files, there may be a hardware or network issue that will reappear over time.

8.5.2.8.2 Additional Shutdown Information The Administration Console generates SNMP messages for managed WebLogic Server instances only until the ServerShutDown message is received. Afterwards, no additional messages are generated.

8.5.2.9 sipAppDeployed

Oracle WebLogic Communication Services engine tier nodes generate this alarm when a SIP Servlet is deployed to the container.

8.5.2.9.1 Recovery Procedure This trap is generated during normal deployment operations and does not indicate an exception.

8.5.2.10 sipAppUndeployed

Oracle WebLogic Communication Services engine tier nodes generate this alarm when a SIP application shuts down, or if a SIP application is undeployed. This generally occurs when Oracle WebLogic Communication Services is shutdown while active requests still exist.

8.5.2.10.1 Recovery Procedure During normal shutdown procedures this alarm should be filtered out and should not reach operations. If the alarm occurs during the course of normal operations, it indicates that someone has shutdown the application or server unexpectedly, or there is a problem with the application. Notify Tier 4 support immediately.

8.5.2.11 sipAppFailedToDeploy

Oracle WebLogic Communication Services engine tier nodes generate this trap when an application deploys successfully as a Web Application but fails to deploy as a SIP application.

8.5.2.11.1 Recovery Procedure The typical failure is caused by an invalid `sip.xml` configuration file and should occur only during software installation or upgrade procedures. When it occurs, undeploy the application, validate the sip.xml file, and retry the deployment.

Note: This alarm should never occur during normal operations. If it does, contact Tier 4 support immediately.

8.6 Using the WebLogic Diagnostics Framework (WLDF)

The WebLogic Diagnostic Framework (WLDF) consists of a number of components that work together to collect, archive, and access diagnostic information about a WebLogic Server instance and its applications. Oracle WebLogic Communication Services version integrates with several components of the WLDF in order to monitor and diagnose the operation of engine and SIP data tier nodes, as well as deployed SIP Servlets:

- Data Collectors—Oracle WebLogic Communication Services integrates with the Harvester service to collect information from runtime MBeans, and with the Logger service to archive SIP requests and responses.
- Watches and Notifications—Administrators can use the Watches and Notifications component to create complex rules, based on Oracle WebLogic Communication Services runtime MBean attributes, that trigger automatic notifications using JMS, JMX, SNMP, SMTP, and so forth.
- Image Capture—Oracle WebLogic Communication Services instances can collect certain diagnostic data and write the data to an image file when requested by an Administrator. This data can then be used to diagnose problems in a running server.
- Instrumentation—Oracle WebLogic Communication Services instruments the server and application code with monitors to help you configure diagnostic actions that are performed on SIP messages (requests and responses) that match certain criteria.

The sections that follow provide more details about how Oracle WebLogic Communication Services integrates with each of the above WLDF components.

8.6.1 Data Collection and Logging

Oracle WebLogic Communication Services uses the WLDF Harvester service to collect data from the attributes of these runtime MBeans:

- `ReplicaRuntimeMBean`
- `SipApplicationRuntimeMBean`
- `SipServerRuntimeMBean`

You can add charts and graphs of this data to your own custom views using the WLDF console extension. To do so, first enable the WLDF console extension by copying the JAR file into the `console-ext` subdirectory of your domain directory:

```
cp ~/bea/wlserver_10.3/server/lib/console-ext/diagnostics-console-extension.jar
~/bea/user_projects/domains/mydomain/console-ext
```

When accessing the WLDF console extension, the Oracle WebLogic Communication Services runtime MBean attributes are available in the Metrics tab of the extension.

Oracle WebLogic Communication Services also uses the WLDF Logger service to archive SIP and Diameter messages to independent, dedicated log files (by default, *domain_home*/logs/*server_name*/sipMessages.log). You can configure the name and location of the log file, as well as log rotation policies, using the Configuration > Message Debug tab in the SIP Server Administration Console extension. Note that a server restart is necessary in order to initiate independent logging and log rotation.

8.6.2 Watches and Notifications

The data collected from Oracle WebLogic Communication Services runtime MBeans can be used to create automated monitors, or "watches," that observe a server's diagnostic state. One or more notifications can then be configured for use by a watch, in order to generate a message using SMTP, SNMP, JMX, or JMS when your configured watch conditions and rules occur.

To use watches and notifications, you select the Diagnostics > Diagnostic Modules node in the left pane of the Administration Console and create a new module with the watch rules and notifications required for monitoring your servers. The watch rules can use the metrics collected from Oracle WebLogic Communication Services runtime MBeans, messages written to the log file, or events generated by the diagnostic framework.

8.6.3 Image Capture

Oracle WebLogic Communication Services adds its own image capture information to the diagnostic image generated by the WLDF. You can generate diagnostic images either on demand, or automatically by configuring watch rules.

The information contained in diagnostic images is intended for use by Oracle technical support personnel when troubleshooting a potential server problem and includes:

- SIP data tier partition and replica configuration
- Call state and timer statistics
- Work manager statistics

8.6.4 Instrumentation

The WLDF instrumentation system creates diagnostic monitors and inserts them into Oracle WebLogic Communication Services or application code at specific points in the flow of execution. Oracle WebLogic Communication Services integrates with the instrumentation service to provide a built-in DyeInjection monitor. When enabled, this monitor injects dye flags into the diagnostic context when certain SIP messages enter or exist the system. Dye flags are injected based on the monitor's configuration properties, and on certain request attributes.

Oracle WebLogic Communication Services adds the dye flags described in Table 8–4 below, as well as the WebLogic Server dye flags USER and ADDR. See *Oracle Fusion Middleware Configuring and Using the Diagnostics Framework for Oracle WebLogic Server* for more information.

Table 8–4 Oracle WebLogic Communication Services DyeInjection Flags

Dye Flag	Description
PROTOCOL_SIP	Set in the diagnostic context of all SIP protocol messages.
SIP_REQ	Set in the diagnostic context for all SIP requests that match the value of the property SIP_REQ.
SIP_RES	Sset if the SIP response matches the value of property SIP_RES.
SIP_REQURI	Set if the SIP request's reqURI matches the value of property SIP_REQURI.
SIP_ANY_HEADER	Set if the SIP request contains a header that matches the value of the property SIP_ANY_HEADER.
SIP_RES	This flag is set in the diagnostic context for all SIP responses that match the value of the property SIP_RES.
SIP_REQURI	This flag is set if a SIP request's request URI matches the value of property SIP_REQURI.
SIP_ANY_HEADER	This flag is set if a SIP request contains a header matching the value of the property SIP_ANY_HEADER. The value of SIP_ANY_HEADER is specified using the format *messageType*.*headerName*=headerValue where *headerValue* is either a value or regular expression. For example, you can specify the property as SIP_ANY_HEADER=request.Contact=sip:sipp@localhost:5061 or SIP_ANY_HEADER=response.Contact=sip:findme@172.17.30.50:5060.

Dye flags can be applied to both incoming and outbound SIP messages. The flags are useful for dye filtering, and can be used by delegating monitors to trigger further diagnostic actions.

Oracle WebLogic Communication Services provides several delegating monitors that can be applied at the application and server scope, and which may examine dye flags set by the `DyeInjection` monitor. The delegating monitors are described in Table 8–4.

Table 8–5 Oracle WebLogic Communication Services Diagnostic Monitors

Monitor Name	Monitor Type	Scope	Pointcuts
occas/Sip_Servlet_Before_Service	Before	Application	At entry of `SipServlet.do*` or `SipServlet.service` methods of all implementing subclasses.
occas/Sip_Servlet_After_Service	After	Application	At exit of `SipServlet.do*` or `SipServlet.service` methods of all implementing subclasses.
occas/Sip_Servlet_Around_Service	Around	Application	At entry and exit of `SipServlet.do*` or `SipServlet.service` methods of all implementing subclasses.
occas/Sip_Servlet_Before_Session	Before	Application	At entry of `getAttribute`, `set`, `remove`, and `invalidate` methods for both `SipSession` and `SipApplicationSession`.
occas/Sip_Servlet_After_Session	After	Application	At exit of `getAttribute`, `set`, `remove`, and `invalidate` methods for both `SipSession` and `SipApplicationSession`.
occas/Sip_Servlet_Around_Session	Around	Application	At entry and exit of `getAttribute`, `set`, `remove`, and `invalidate` methods for both `SipSession` and `SipApplicationSession`.

Table 8–5 (Cont.) Oracle WebLogic Communication Services Diagnostic Monitors

Monitor Name	Monitor Type	Scope	Pointcuts
occas/SipSessionDebug	Around	Application	This is a built-in, application-scoped monitor having fixed pointcuts and a fixed debug action. Before and after a pointcut, the monitor performs the `SipSessionDebug` diagnostic action, which calculates the size of the SIP session after serializing the underlying object. The pointcuts for this monitor are as follows: 1. Before and after calls to `getSession` and `getApplicationSession` of the `SipServletMessage` class hierarchy. 2. Before and after calls to `getAttribute`, `setAttribute`, and `removeAttribute` methods in the `SipSession` and `SipApplicationSession` classes. **Note:** The `occas/SessionDebugAction-Before` event is not triggered for the `req.getSession()` or `req.getApplicationSession()` joinpoints. Only the `occas/SessionDebugAction-After` is triggered, because the Session is made available for inspection only after the joinpoints have executed. **Note:** If you compile your application using Apache Ant, you must enable the `debug` attribute to embed necessary debug information into the generated class files.
occas/Sip_Servlet_Before_Message_Send_Internal	Before	Server	At entry of Oracle WebLogic Communication Services code that writes messages to the wire.
occas/Sip_Servlet_After_Message_Send_Internal	After	Server	At exit of Oracle WebLogic Communication Services code that writes messages to the wire.
occas/Sip_Servlet_Around_Message_Send_Internal	Around	Server	At entry and exit of Oracle WebLogic Communication Services code that writes messages to the wire.

8.6.4.1 Configuring Server-Scoped Monitors

To use the server-scoped monitors, you must create a new diagnostic module and create and configure one or more monitors in the module. For the built-in DyeInjection monitor, you then add monitor properties to define the specific dye flags. For delegating monitors such as `occas/Sip_Servlet_Before_Message_Send_Internal`, you add monitor properties to define diagnostic actions.

Follow these steps to configure the Oracle WebLogic Communication Services DyeInjection monitor, a delegate monitor, and enable dye filtering:

1. Access the Administration Console for you domain.
2. Select the Diagnostics > Diagnostic Modules node in the left pane of the console.
3. Click New to create a new Diagnostic Module. Give the module a descriptive name, such as "instrumentationModule," and click OK.
4. Select the new "instrumentationModule" from the list of modules in the table.
5. Select the Targets tab.

6. Select a server on which to target the module and click Save.
7. Return to the Diagnostics > Diagnostic Modules node and select instrumentationModule from the list of modules.
8. Select the Configuration > Instrumentation tab.
9. Select Enabled to enable instrumentation at the server level, then click Save.
10. Add the DyeInjection monitor to the module:
    a. Click Add/Remove.
    b. Select the name of a monitor from the Available list (for example, DyeInjection), and use the arrows to move it to the Chosen list.
    c. Click OK.
    d. Select the newly-created monitor from the list of available monitors.
    e. Ensure that the monitor is enabled, and edit the Properties field to add any required properties. For the DyeInjection monitor, sample properties include:

```
SIP_RES=180
SIP_REQ=INVITE
SIP_ANY_HEADER=request.Contact=sip:sipp@localhost:5061
```

    f. Click Save
11. Add one or more delegate monitors to the module:
    a. Return to the Configuration > Instrumentation tab for the new module.
    b. Click Add/Remove.
    c. Select the name of a delegate monitor from the Available list (for example, `occas/Sip_Servlet_Before_Message_Send_Internal`), and use the arrows to move it to the Chosen list.
    d. Click OK.
    e. Select the newly-created monitor from the list of available monitors.
    f. Ensure that the monitor is enabled, then select one or more Actions from the available list, and use the arrows to move the actions to the Chosen list. For the `occas/Sip_Servlet_Before_Message_Send_Internal` monitor, sample actions include `DisplayArgumentsAction`, `StackDumpAction`, `ThreadDumpAction`, and `TraceAction`.
    g. Select the check box to EnableDyeFiltering.
    h. Select one or more Dye Masks, such as SIP_REQ, from the Available list and use the arrows to move them to the Chosen list.
    i. Click Save

Note: You can repeat the above steps to create additional delegate monitors.

8.6.4.2 Configuring Application-Scoped Monitors

You configure application-scoped monitors in an XML configuration file named `weblogic-diagnostics.xml`. You must store the `weblogic-diagnostics.xml` file in the SIP module's or enterprise application's `META-INF` directory.

The XML file enables instrumentation at the application level, defines point cuts, and also defines delegate monitor dye masks and actions. Example 8–1 shows a sample configuration file that uses the `occas/Sip_Servlet_Before_Service` monitor.

Example 8–1 Sample weblogic-diagnostics.xml File

```
<wldf-resource xmlns="http://www.bea.com/ns/weblogic/90/diagnostics">
  <instrumentation>
    <enabled>true</enabled>
    <include>demo.ProxyServlet</include>
    <wldf-instrumentation-monitor>
      <name>occas/Sip_Servlet_Before_Service</name>
      <enabled>true</enabled>
      <dye-mask>SIP_ANY_HEADER</dye-mask>
      <dye-filtering-enabled>true</dye-filtering-enabled>
      <action>DisplayArgumentsAction</action>
    </wldf-instrumentation-monitor>
   </instrumentation>
</wldf-resource>
```

In this example, if an incoming request's diagnostic context contains the `SIP_ANY_HEADER` dye flag, then the `occas/Sip_Servlet_Before_Service` monitor is triggered and the `DisplayArgumentsAction` is executed.

8.7 Logging SIP Requests and Responses

Oracle WebLogic Communication Services enables you to perform Protocol Data Unit (PDU) logging for the SIP requests and responses it processes. Logged SIP messages are placed either in the domain-wide log file for Oracle WebLogic Communication Services, or in the log files for individual Managed Server instances. Because SIP messages share the same log files as Oracle WebLogic Communication Services instances, you can use advanced server logging features such as log rotation, domain log filtering, and maximum log size configuration when managing logged SIP messages.

Administrators configure SIP PDU logging by defining one or more SIP Servlets using the `com.bea.wcp.sip.engine.tracing.listener.TraceMessageListenerImpl` class. Logging criteria are then configured either as parameters to the defined servlet, or in separate XML files packaged with the application.

As SIP requests are processed or SIP responses generated, the logging Servlet compares the message with the filtering patterns defined in a standalone XML configuration file or Servlet parameter. SIP requests and responses that match the specified pattern are written to the log file along with the name of the logging servlet, the configured logging level, and other details. To avoid unnecessary pattern matching, the Servlet marks new SIP Sessions when an initial pattern is matched and then logs subsequent requests and responses for that session automatically.

Logging criteria are defined either directly in `sip.xml` as parameters to a logging Servlet, or in external XML configuration files. See Section 8.7.3, "Specifying the Criteria for Logging Messages".

Note: Engineers can implement PDU logging functionality in their Servlets either by creating a delegate with the `TraceMessageListenerFactory` in the Servlet's `init()` method, or by using the tracing class in deployed Java applications. Using the delegate enables you to perform custom logging or manipulate incoming SIP messages using the default trace message listener implementation. See Section 8.7.7, "Adding Tracing Functionality to SIP Servlet Code" for an example of using the factory in a Servlet's `init()` method.

8.7.1 Defining Logging Servlets in sip.xml

Logging Servlets for SIP messages are created by defining Servlets having the implementation class `com.bea.wcp.sip.engine.tracing.listener.TraceMessageListenerImpl`. The definition for a sample `msgTraceLogger` is shown in Example 8–2.

Example 8–2 Sample Logging Servlet

```
<servlet>
    <servlet-name>msgTraceLogger</servlet-name>

<servlet-class>com.bea.wcp.sip.engine.tracing.listener.TraceMessageListenerImpl</s
ervlet-class>
    <init-param>
      <param-name>domain</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>level</param-name>
      <param-value>full</param-value>
    </init-param>
    <load-on-startup/>
  </servlet>
```

8.7.2 Configuring the Logging Level and Destination

Logging attributes such as the level of logging detail and the destination log file for SIP messages are passed as initialization parameters to the logging Servlet. Table 8–2 lists the parameters and parameter values that you can specify as `init-param` entries. Example 8–2 shows the sample `init-param` entries for a Servlet that logs full SIP message information to the domain log file.

8.7.3 Specifying the Criteria for Logging Messages

The criteria for selecting SIP messages to log can be defined either in XML files that are packaged with the logging Servlet's application, or as initialization parameters in the Servlet's `sip.xml` deployment descriptor. The sections that follow describe each method.

8.7.3.1 Using XML Documents to Specify Logging Criteria

If you do not specify logging criteria as an initialization parameter to the logging Servlet, the Servlet looks for logging criteria in a pair of XML descriptor files in the top level of the logging application. These descriptor files, named `request-pattern.xml` and `response-pattern.xml`, define patterns that Oracle

WebLogic Communication Services uses for selecting SIP requests and responses to place in the log file.

> **Note:** By default Oracle WebLogic Communication Services logs both requests and responses. If you do not want to log responses, you must define a `response-pattern.xml` file with empty matching criteria.

A typical pattern definition defines a condition for matching a particular value in a SIP message header. For example, the sample `response-pattern.xml` used by the `msgTraceLogger` Servlet matches all MESSAGE requests. The contents of this descriptor are shown in

Example 8–3 Sample response-pattern.xml for msgTraceLogger Servlet

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pattern
   PUBLIC "Registration//Organization//Type Label//Definition Language"
   "trace-pattern.dtd">
<pattern>
  <equal>
    <var>response.method</var>
    <value>MESSAGE</value>
  </equal>
</pattern>
```

Additional operators and conditions for matching SIP messages are described in Section 8.7.6, "trace-pattern.dtd Reference". Most conditions, such as the `equal` condition shown in Example 8–3, require a variable (`var` element) that identifies the portion of the SIP message to evaluate. Table 8–2 lists some common variables and sample values. For additional variable names and examples, see *Section 16: Mapping Requests to Servlets* in the SIP Servlet API 1.1 specification (`http://jcp.org/en/jsr/detail?id=289`); Oracle WebLogic Communication Services enables mapping of both request and response variables to logging Servlets.

Table 8–6 Pattern-matching Variables and Sample Values

Variable	Sample Values
request.method, response.method	MESSAGE, INVITE, ACK, BYE, CANCEL
request.uri.user, response.uri.user	guest, admin, joe
request.to.host, response.to.host	server.mydomain.com

Both `request-pattern.xml` and `response-pattern.xml` use the same Document Type Definition (DTD). See Section 8.7.6, "trace-pattern.dtd Reference" for more information.

8.7.3.2 Using Servlet Parameters to Specify Logging Criteria

Pattern-matching criteria can also be specified as initialization parameters to the logging Servlet, rather than as separate XML documents. The parameter names used to specify matching criteria are `request-pattern-string` and `response-pattern-string`. They are defined along with the logging level and destination as described in Section 8.7.2, "Configuring the Logging Level and Destination".

The value of each pattern-matching parameter must consist of a valid XML document that adheres to the DTD for standalone pattern definition documents (see Section 8.7.3.1, "Using XML Documents to Specify Logging Criteria"). Because the XML documents that define the patterns and values must not be parsed as part of the `sip.xml` descriptor, you must enclose the contents within the `CDATA` tag. Example 8–4 shows the full `sip.xml` entry for the sample logging Servlet, `invTraceLogger`. The final two `init-param` elements specify that the Servlet log only `INVITE` request methods and `OPTIONS` response methods.

Example 8–4 Logging Criteria Specified as init-param Elements

```
<servlet>
      <servlet-name>invTraceLogger</servlet-name>

<servlet-class>com.bea.wcp.sip.engine.tracing.listener.TraceMessageListenerImpl</s
ervlet-class>
      <init-param>
        <param-name>domain</param-name>
        <param-value>true</param-value>
      </init-param>
      <init-param>
        <param-name>level</param-name>
        <param-value>full</param-value>
      </init-param>
      <init-param>
        <param-name>request-pattern-string</param-name>
        <param-value>
            <![CDATA[
                <?xml version="1.0" encoding="UTF-8"?>
                <!DOCTYPE pattern
                   PUBLIC "Registration//Organization//Type Label//Definition
Language"
                   "trace-pattern.dtd">
                <pattern>
                  <equal>
                    <var>request.method</var>
                    <value>INVITE</value>
                  </equal>
                </pattern>
            ]]>
        </param-value>
      </init-param>
      <init-param>
        <param-name>response-pattern-string</param-name>
        <param-value>
            <![CDATA[
                <?xml version="1.0" encoding="UTF-8"?>
                <!DOCTYPE pattern
                   PUBLIC "Registration//Organization//Type Label//Definition
Language"
                   "trace-pattern.dtd">
                <pattern>
                  <equal>
                    <var>response.method</var>
                    <value>OPTIONS</value>
                  </equal>
                </pattern>
            ]]>
        </param-value>
      </init-param>
```

```
    <load-on-startup/>
</servlet>
```

8.7.4 Specifying Content Types for Unencrypted Logging

By default Oracle WebLogic Communication Services uses String format (UTF-8 encoding) to log the content of SIP messages having a text or application/sdp Content-Type value. For all other Content-Type values, Oracle WebLogic Communication Services attempts to log the message content using the character set specified in the `charset` parameter of the message, if one is specified. If no `charset` parameter is specified, or if the `charset` value is invalid or unsupported, Oracle WebLogic Communication Services uses Base-64 encoding to encrypt the message content before logging the message.

If you want to avoid encrypting the content of messages under these circumstances, specify a list of String-representable Content-Type values using the `string-rep` element in `sipserver.xml`. The `string-rep` element can contain one or more `content-type` elements to match. If a logged message matches one of the configured `content-type` elements, Oracle WebLogic Communication Services logs the content in String format using UTF-8 encoding, regardless of whether or not a `charset` parameter is included.

> **Note:** You do not need to specify text/* or application/sdp content types as these are logged in String format by default.

Example 8–5 shows a sample `message-debug` configuration that logs String content for three additional Content-Type values, in addition to text/* and application/sdp content.

Example 8–5 Logging String Content for Additional Content Types

```
<message-debug>
  <level>full</level>
  <string-rep>
    <content-type>application/msml+xml</content-type>
    <content-type>application/media_control+xml</content-type>
    <content-type>application/media_control</content-type>
  </string-rep>
</message-debug>
```

8.7.5 Enabling Log Rotation and Viewing Log Files

The Oracle WebLogic Communication Services logging infrastructure enables you to automatically write to a new log file when the existing log file reaches a specified size. You can also view log contents using the Administration Console or configure additional server-level events that are written to the log.

8.7.6 trace-pattern.dtd Reference

`trace-pattern.dtd` defines the required contents of the `request-pattern.xml` and `response-pattern.xml`, documents, as well as the values for the `request-pattern-string` and `response-pattern-string` Servlet `init-param` variables.

Example 8–6 trace-pattern.dtd

```
<!--
The different types of conditions supported.
- >

<!ENTITY % condition "and | or | not |
                     equal | contains | exists | subdomain-of">

<!--
A pattern is a condition: a predicate over the set of SIP requests.
- >

<!ELEMENT pattern (%condition;)>

<!--
An "and" condition is true if and only if all its constituent conditions
are true.
- >

<!ELEMENT and (%condition;)+>

<!--
An "or" condition is true if at least one of its constituent conditions
is true.
- >

<!ELEMENT or (%condition;)+>

<!--
Negates the value of the contained condition.
- >

<!ELEMENT not (%condition;)>

<!--
True if the value of the variable equals the specified literal value.
- >

<!ELEMENT equal (var, value)>

<!--
True if the value of the variable contains the specified literal value.
- >

<!ELEMENT contains (var, value)>

<!--
True if the specified variable exists.
- >

<!ELEMENT exists (var)>

<!--
- >

<!ELEMENT subdomain-of (var, value)>

<!--
Specifies a variable. Example:
  <var>request.uri.user</var>
```

```
- >

<!ELEMENT var (#PCDATA)>

<!--
Specifies a literal string value that is used to specify rules.
- >

<!ELEMENT value (#PCDATA)>

<!--
Specifies whether the "equal" test is case sensitive or not.
- >

<!ATTLIST equal ignore-case (true|false) "false">

<!--
Specifies whether the "contains" test is case sensitive or not.
- >

<!ATTLIST contains ignore-case (true|false) "false">

<!--
The ID mechanism is to allow tools to easily make tool-specific
references to the elements of the deployment descriptor. This allows
tools that produce additional deployment information (i.e information
beyond the standard deployment descriptor information) to store the
non-standard information in a separate file, and easily refer from
these tools-specific files to the information in the standard sip-app
deployment descriptor.
- >

<!ATTLIST pattern id ID #IMPLIED>
<!ATTLIST and id ID #IMPLIED>
<!ATTLIST or id ID #IMPLIED>
<!ATTLIST not id ID #IMPLIED>
<!ATTLIST equal id ID #IMPLIED>
<!ATTLIST contains id ID #IMPLIED>
<!ATTLIST exists id ID #IMPLIED>
<!ATTLIST subdomain-of id ID #IMPLIED>
<!ATTLIST var id ID #IMPLIED>
<!ATTLIST value id ID #IMPLIED>
```

8.7.7 Adding Tracing Functionality to SIP Servlet Code

Tracing functionality can be added to your own Servlets or to Java code by using the `TraceMessageListenerFactory`. `TraceMessageListenerFactory` enables clients to reuse the default trace message listener implementation behaviors by creating an instance and then delegating to it. The factory implementation instance can be found in the servlet context for SIP Servlets by looking up the value of the `TraceMessageListenerFactory.TRACE_MESSAGE_LISTENER_FACTORY` attribute.

> **Note:** Instances created by the factory are not registered with Oracle WebLogic Communication Services to receive callbacks upon SIP message arrival and departure.

To implement tracing in a Servlet, you use the factory class to create a delegate in the Servlet's `init()` method as shown in Example 8–7.

Example 8–7 Using the TraceMessageListenerFactory

```
public final class TraceMessageListenerImpl extends SipServlet implements
MessageListener {
  private MessageListener delegate;

  public void init() throws ServletException {
    ServletContext sc = (ServletContext) getServletContext();
    TraceMessageListenerFactory factory = (TraceMessageListenerFactory)
sc.getAttribute(TraceMessageListenerFactory.TRACE_MESSAGE_LISTENER_FACTORY);
    delegate = factory.createTraceMessageListener(getServletConfig());
  }
  public final void onRequest(SipServletRequest req, boolean incoming) {
    delegate.onRequest(req,incoming);
  }
  public final void onResponse(SipServletResponse resp, boolean incoming) {
    delegate.onResponse(resp,incoming);
  }
}
```

8.7.8 Order of Startup for Listeners and Logging Servlets

If you deploy both listeners and logging servlets, the listener classes are loaded first, followed by the Servlets. Logging Servlets are deployed in order according to the load order specified in their Web Application deployment descriptor.

8.8 Tuning JVM Garbage Collection for Production Deployments

Production installations of Oracle WebLogic Communication Services generally require extremely small response times (under 50 milliseconds) for clients at all times, even under peak server loads. A key factor in maintaining brief response times is the proper selection and tuning of the JVM's Garbage Collection (GC) algorithm for Oracle WebLogic Communication Services instances in the engine tier.

Whereas certain tuning strategies are designed to yield the lowest average garbage collection times or to minimize the frequency of full GCs, those strategies can sometimes result in one or more very long periods of garbage collection (often several seconds long) that are offset by shorter GC intervals. With a production SIP Server installation, all long GC intervals must be avoided in order to maintain response time goals.

The sections that follow describe GC tuning strategies for JRockit and Sun's JVM that generally result in best response time performance.

Note: For more information on JRockit, see *Oracle Fusion Middleware Introduction to Oracle WebLogic Server*.

8.8.1 Modifying JVM Parameters in Server Start Scripts

If you use custom startup scripts to start Oracle WebLogic Communication Services engines and replicas, simply edit those scripts to include the recommended JVM options described in the sections that follow.

The Configuration Wizard also installs default startup scripts when you configure a new domain. These scripts are installed in the `MIDDLEWARE_HOME/user_projects/domains/domain_name/bin` directory by default, and include:

- `startWebLogic.cmd`, `startWebLogic.sh`—These scripts start the Administration Server for the domain.
- `startManagedWebLogic.cmd`, `startManagedWebLogic.sh`—These scripts start managed engines and replicas in the domain.

If you use the Oracle-installed scripts to start engines and replicas, you can override JVM memory arguments by first setting the `USER_MEM_ARGS` environment variable in your command shell.

Note: Setting the `USER_MEM_ARGS` environment variable overrides all default JVM memory arguments specified in the Oracle-installed scripts. Always set `USER_MEM_ARGS` to the full list of JVM memory arguments you intend to use. For example, when using the Sun JVM, always add `-XX:MaxPermSize=128m` to the `USER_MEM_ARGS` value, even if you only intend to change the default heap space (`-Xms`, `-Xmx`) parameters.

8.8.2 Tuning Garbage Collection with JRockit

JRockit provides several monitoring tools that you can use to analyze the JVM heap at any given moment, including:

- JRockit Runtime Analyzer—provides a view into the runtime behavior of garbage collection and pause times.
- JRockit Stack Dumps—reveals applications' thread activity to help you troubleshoot and/or improve performance.

Use these and other tools in a controlled environment to determine the effects of JVM settings before you use the settings in a production deployment.

The following sections describe suggested starting JVM options for use with the JRockit. If you use JRockit with the deterministic garbage collector (recommended), use the options described in Section 8.8.3, "Using Oracle JRockit Real Time (Deterministic Garbage Collection)".

8.8.3 Using Oracle JRockit Real Time (Deterministic Garbage Collection)

Very short response times are most easily achieved by using JRockit Real Time, which implements a deterministic garbage collector.

Oracle recommends using the following JVM arguments for engine tier servers in replicated cluster configurations:

```
-Xms1024m -Xmx1024m -XgcPrio:deterministic -XpauseTarget=30ms -XXtlasize:min=8k
-XXnosystemgc
```

> **Note:** The above settings are configured by default in the `$WLSS_HOME/common/bin/wlssCommenv.sh` file when you use the Configuration Wizard to create a new domain with the JRockit JVM.
>
> You may need to increase the `-XpauseTarget` value for allocation-intensive applications. The value can be decreased for smaller applications under light loads.
>
> Adjust the heap size according to the amount of live data used by deployed applications. As a starting point, set the heap size from 2 to 3 times the amount required by your applications. A value closer to 3 times the required amount generally yields the best performance.

For replica servers, increase the available memory:

```
-Xms3072m -Xmx3072m -XgcPrio:deterministic -XpauseTarget=30ms -XXtlasize:min=8k
-XXnosystemgc
```

These settings fix the heap size and enable the dynamic garbage collector with deterministic garbage collection. `-XpauseTarget` sets the maximum pause time and `-XXtlasize=3k` sets the thread-local area size. `-XXnosystemgc` prevents `System.gc()` application calls from forcing garbage collection.

8.8.4 Using Oracle JRockit without Deterministic Garbage Collection

When using Oracle's JRockit JVM without deterministic garbage collection (not recommended for production deployments), the best response time performance is obtained by using the generational concurrent garbage collector.

The full list of example startup options for an engine tier server are:

```
-Xms1024m -Xmx1024m -Xgc:gencon -XXnosystemgc -XXtlasize:min=3k -XXkeeparearatio=0
-Xns:48m
```

> **Note:** Fine tune the heap size according to the amount of live data used by deployed applications.

The full list of example startup options for a replica server are:

```
-Xms3072m -Xmx3072m -Xgc:gencon -XXnosystemgc -XXtlasize:min=3k -XXkeeparearatio=0
-Xns:48m
```

8.8.5 Tuning Garbage Collection with Sun JDK

When using Sun's JDK, the goal in tuning garbage collection performance is to reduce the time required to perform a full garbage collection cycle. You should not attempt to tune the JVM to minimize the frequency of full garbage collections, because this generally results in an eventual forced garbage collection cycle that may take up to several full seconds to complete.

The simplest and most reliable way to achieve short garbage collection times over the lifetime of a production server is to use a fixed heap size with the default collector and the parallel young generation collector, restricting the new generation size to at most one third of the overall heap.

The following example JVM settings are recommended for most engine tier servers:

```
-server -Xmx1024m -XX:MaxPermSize=128m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
```

```
-XX:+UseTLAB -XX:+CMSIncrementalMode -XX:+CMSIncrementalPacing
-XX:CMSIncrementalDutyCycleMin=0 -XX:CMSIncrementalDutyCycle=10
-XX:MaxTenuringThreshold=0 -XX:SurvivorRatio=256
-XX:CMSInitiatingOccupancyFraction=60 -XX:+DisableExplicitGC
```

For replica servers, use the example settings:

```
-server -Xmx3072m -XX:MaxPermSize=128m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-XX:+UseTLAB -XX:+CMSIncrementalMode -XX:+CMSIncrementalPacing
-XX:CMSIncrementalDutyCycleMin=0 -XX:CMSIncrementalDutyCycle=10
-XX:MaxTenuringThreshold=0 -XX:SurvivorRatio=256
-XX:CMSInitiatingOccupancyFraction=60 -XX:+DisableExplicitGC
```

The above options have the following effect:

- `-XX:+UseTLAB`—Uses thread-local object allocation blocks. This improves concurrency by reducing contention on the shared heap lock.
- `-XX:+UseParNewGC`—Uses a parallel version of the young generation copying collector alongside the concurrent mark-and-sweep collector. This minimizes pauses by using all available CPUs in parallel. The collector is compatible with both the default collector and the Concurrent Mark and Sweep (CMS) collector.
- `-Xms, -Xmx`—Places boundaries on the heap size to increase the predictability of garbage collection. The heap size is limited in replica servers so that even Full GCs do not trigger SIP retransmissions. `-Xms` sets the starting size to prevent pauses caused by heap expansion.
- `-XX:MaxTenuringThreshold=0`—Makes the full NewSize available to every NewGC cycle, and reduces the pause time by not evaluating tenured objects. Technically, this setting promotes all live objects to the older generation, rather than copying them.
- `-XX:SurvivorRatio=128`—Specifies a high survivor ratio, which goes along with the zero tenuring threshold to ensure that little space is reserved for absent survivors.

8.9 Avoiding JVM Delays Caused By Random Number Generation

The library used for random number generation in Sun's JVM relies on `/dev/random` by default for UNIX platforms. This can potentially block the Oracle WebLogic Communication Services process because on some operating systems `/dev/random` waits for a certain amount of "noise" to be generated on the host machine before returning a result. Although `/dev/random` is more secure, Oracle recommends using `/dev/urandom` if the default JVM configuration delays Oracle WebLogic Communication Services startup.

To determine if your operating system exhibits this behavior, try displaying a portion of the file from a shell prompt:

```
head -n 1 /dev/random
```

1. Open the `$JAVA_HOME/jre/lib/security/java.security` file in a text editor.
2. Change the line:

   ```
   securerandom.source=file:/dev/random
   ```

   to read:

   ```
   securerandom.source=file:/dev/urandom
   ```

3. Save your change and exit the text editor.

Part II

Configuring Presence

This Part contains the following chapter:

9

Configuring Presence and Presence Web Services

This chapter provides an introduction to the Oracle WebLogic Communication Services (OWLCS) in the following sections:

9.1 Overview of Presence

Presence may be used to display an end-user's availability and ability to participate in a chat or richer multimedia interaction. Client presence is often represented as a contact management list, which displays user availability as icons. These icons, which not only represent a user's availability, but also a user's location, means of contact, or current activity, enable efficient communications between users.

The Presence application enables a service provider or an enterprise to extend presence service to end users. Major capabilities include:

- Presence Status Publication
- Presence Status Subscriptions
- Privacy
- Presence Hard State

Presence Status Publication

The term *presentity* is used here to refer to a *Presence Entity* (a Presence Entity [presentity] is an entity, such as a person, who is defined by their ability and willingness to communicate). A presentity can publish a Presence Information Data Format (PIDF) document containing presence state to the Presence Server.

Presence Status Subscriptions

The Presence Server supports subscriptions to a presentity's (that is, a user's) status. The Presence Server supports the *watcher information event package*. This event package allows a presentity to be notified as soon as a watcher has subscribed to his/her presence state. The Presence Server notifies the user when the watcher (subscriber) requests authorization to view the presentity's status. The Presence server also notifies all of the active, authorized watchers of the publication of a new presence document.

Privacy

A presentity can create filtering rules allowing certain watchers to only see certain parts of the presence states. Whenever a watcher subscribes to a presentity's presence, the Presence Server checks the authorization policy that the presentity has set to see if the watcher has the required authorization.

If no matching rule can be found, the watcher is put in a pending state and a watcher info notification is sent to the presentity. Usually, the presentity's client (User Agent) presents a pop-up box asking whether to accept or reject a new pending watcher. The answer is added to the presentity's authorization policy document in the form of a rule for this watcher. The document is then updated by the client on the XDMS using HTTP. When the document is updated, the Presence Server reads the new policy document and acts on the new rule, changing the subscription state accordingly.

Presence Hard State

A presentity can leave a hard state note about their presence (such as when going on vacation: 'On vacation - back on the 15th'). Watchers to this presentity's presence state would be able to see this information.

9.2 Configuring Presence

Configuration of the Presence Server is done through the following MBeans:

- Section 9.2.1, "Configuring XDMS"
- Section 9.2.2, "Bus"
- Section 9.2.3, "PackageManager"
- Section 9.2.4, "Presence"
- Section 9.2.5, "PresenceEventPackage"
- Section 9.2.6, "PresenceWInfoEventPackage"
- Section 9.2.7, "UA-ProfileEventPackage"
- Section 9.2.8, "Command Service (XDMS Provisioning)"
- Section 9.2.9, "XCapConfigManager"
- Section 9.2.10, "Aggregation Proxy"
- Section 9.2.11, "Configuring Default Application Router for OPTIONS"

9.2.1 Configuring XDMS

The following MBeans enables you to configure the XDMS (XML Document Management Server):

- Command Service (XDMS Provisioning)
- XCapConfigManager

Note: If you change any attributes of the following MBeans, you must restart OWLCS for these changes to take effect.

- Presence
- PresenceEventPackage
- PresenceWInfoEventPackage
- UAProfileEventPackage
- XCapConfigManager

Note: JGroups channels must only communicate with their intended peers. To ensure that event packages use JGroups correctly, different channels must use different values for the group name, multicast address and multicast port. See `http://www.jboss.org/` for more information.

9.2.2 Bus

Through the Bus MBean you can configure the internal asynchronous bus that the Presence Server is using for its internal job execution. Table 9–1 describes the attributes of the Bus MBean.

Table 9–1 Attributes of the Bus MBean

Attribute	Value Type	Description
ThreadPoolSize	int	The number of threads held in the thread pool, which remains constant throughout the lifetime of the application. If no threads are used, then the specified number of threads remain idle. The default value is 15.
HighWatermark	int	The number of pending jobs reached before the bus's exhausted threshold level is reached. The default value is 20.
KeepAlive	long	The number of seconds to keep an idle thread alive before dropping it. The default value is 60.
LogDuration	long	A warning is logged to the system log for events that remain in the queue for a period exceeding the specified duration before they are broadcast to the bus. This warning indicates that server is about to be overloaded, since an old job has been sent to the bus. The default value is 60.
LowWatermark	int	Specifies the low threshold level for the number of pending jobs. When this threshold is reached from below, the Bus logs a warning that it is about to be choked. At this point, no more warnings are logged until the high watermark level is reached. The default value is 15.

9.2.3 PackageManager

The PresenceEventPackage, PresenceWInfoEventPackage, and UA-ProfileEventPackage MBeans enable you to configure the event packages, which define the state information to be reported by a notifier to a watcher (subscriber).

These packages form the core of the Presence Server, as most requests flow through them.

A notifier is a User Agent (UA) that generates NOTIFY requests that alert watchers to the state of a resource (the entity about which watchers request state information). Notifiers typically accept SUBSCRIBE requests to create subscriptions. A watcher is another type of UA, one that receives the NOTIFY requests issued by a notifier. Such requests contain information about the state of a resource of interest to the watcher. Watchers typically also generate SUBSCRIBE requests and send them to notifiers to create subscriptions.

The PackageManager MBean sets the configuration that determines which of the three event packages (Presence, Watcher Info and UA-Profile) get loaded, as well as configures the environment in which these event packages are loaded. Table 9–2 describes the attributes of the PackageManger MBean.

Table 9–2 Attributes of the EventPackageManager MBean

Attribute	Description
JGroupBroadcastEnabled	If true, a single JGroup channel is created for all the event packages running on this Presence Server instance. Whenever new resources are created in any of the running event packages, a message is broadcast on this JGroup channel.
JGroupXMLConfigPath	Path to an XML configuration file for JGroups. The path can be absolute or relative to the WebLogic domain directory on which the event package is running. Leave this empty to use the following default values for the jgroups connection: UDP(bind_addr=[ip address of this host];mcast_addr=230.0.0.1;mcast_port=7426;ip_ttl=1)
JGroupChannelName	The name to use when creating the JGroup channel. Note that to prevent aliasing of different JGroup clusters, each cluster must have a unique channel name in addition to a unique multicast port or address.
CaseSensitiveUserPart	Setting this attribute to *true* enables case-sensitive handling of the user part of the SIP URI. If this parameter is set to *false*, then the user part of the URI is not a case-sensitive match. For example, *foo* is considered the same as *FoO*. The domain part of the URI is always case-insensitive.
EventPackageNames	A comma-separated list of event package names. For example: *presence,presence.winfo,ua-profile*. Only the event packages listed here will be started by the Presence Server.
WaitingSubsCleanupInterval	The interval, in seconds, in which the subscription cleanup check runs. The thread sleeps for this period and then awakens to check for any waiting subscriptions with a timestamp older than the *MaxWaitingSubsTimeHours* parameter. All old subscriptions are then removed from the subscribed resource.
Max WaitingSubsTimeHours	The maximum time, in hours, that a subscription can be in a waiting state before the server removes it. This parameter is used by the subscription cleanup check thread (`waitingsubscleanupinterval`) to decide if a waiting subscription is old enough to be removed from the subscribed resource.

9.2.4 Presence

The Presence MBean controls how the Presence Server interacts with clients connecting to it. The attributes (described in Table 9–3) include those for setting the composition policy for creating a unified document when a presentity publishes

presence documents from two or more clients, as well as setting the blocking, filtering, and presence hard state.

Table 9–3 Attributes of the Presence MBean

Attribute	Description/Value
CompositionPolicyFilename	The filename of the composition policy document. Values include `compose.xslt`, for the OWLCS composition policy, and `compose_OMA.xslt`, for the OMA composition policy.
DefaultSubHandling	The default subscription authorization decision that the server makes when no presence rule is found for an authenticated user. The defined values are: ■ block ■ confirm ■ polite-block Unauthenticated users will always be blocked if no rule is found.
DocumentStorageFactory	The name of the `DocumentStorage` Factory Class. The default value is `oracle.sdp.presenceeventpackage.document.XMLDocumentStorageFactoryImpl`.
DocumentStorageRootUrl	The system identifier for the document storage. In the file storage case, this is the root file URL path where documents are stored. The content of this directory should be deleted when the server is restarted. The default value is *file:/tmp/presencestorage/*.
DocumentStorageType	The type of storage to be used for presence documents. Valid values: ■ file ■ memory The default value is *memory*.
HttpAssertedIdentityHeader	The type of asserted identity header used in all HTTP requests from the Presence Server to the XDMS. Set the value of this attribute to one expected by the XDMS. Valid values: ■ X_3GPP_ASSERTED_IDENTITY ■ X_3GPP_INTENDED_IDENTITY ■ X_XCAP_ASSERTED_IDENTITY (The default value.)
PidfManipulationAuid	Also known as *hard state*, the ID of the application usage for PIDF (Presence Information Data Format) manipulation. The default value is *pidf-manipulation*.
PidfManipulationDocumentName	The document name for pidf manipulation application usage, for example: *hardstate*. The default value is *hardstate*.
PidfManipulationEnabled	Set to *true* (the default value) to enable PIDF manipulation.
PidfManipulationXcapUri	The SIP URI of the XDMS for the pidf manipulation application usage. The default value is: *sip:127.0.0.1;transport=TCP;lr*. The loose route (*lr*) parameter must be included in the SIP URI for the server to function properly.
PoliteBlockPendingSubscription	Set to *true* if pending subscriptions should be polite-blocked. This feature is used to hide the presentity from the presence watcher with a pending subscription and instead send them fake presence documents. If set to *false* the subscriptions will remain as pending.
PresRulesAuid	The ID of the application usage for presrules. The default is *pres-rules*.
PresRulesDocumentName	The document name for presrules application usage. The default value is *presrules*.

Table 9–3 (Cont.) Attributes of the Presence MBean

Attribute	Description/Value
PresRulesXcapUri	The SIP URI of the XDMS for the presence rules application usage. The default value is: *sip:127.0.0.1; transport=TCP;lr*. The loose route (*lr*) parameter must be included in the SIP URI for the server to function properly.
PrivacyFilteringEnabled	Set to *true* to enable privacy filtering. Set to *false* to disable filtering. If privacy filtering is disabled, then all subscriptions that are allowed to see a presentity's presence will always see everything that has been published for the presentity.
TransformerFactory	The name of the `TransformerFactory` class. The default value is `oracle.xml.jaxp.JXSAXTransformerFactory`.

9.2.5 PresenceEventPackage

Table 9–4 shows the attributes of the PresenceEventPackage MBean. The presence event package has two subgroups: publish and subscribe. Each subgroup has a `minexpires` and a `maxexpires` parameter. A client may suggest an expiration time for its subscription or its published state but if the suggested time that is lower than the configured *minExpires*, the server will return a 423 (Interval Too Brief) response.

If a client suggests an expiration time that is higher than the configured max expiration, the server will shorten the suggested time to this configured value. The value chosen by the server is always conveyed in the response to the request.

To keep a publication or subscription alive, the client sends `republish` or `resubscribe` to the server within the expiry time. The client must perform this task repeatedly through the lifetime of the publication or subscription.

Table 9–4 Attributes of the PresenceEventPackage

Attribute	Value/Description
Description	A description of the PresenceEventPackage. For example: *The event package that enables presence*.
DocumentFactory	The `DocumentFactory` class name. The default value is `oracle.sdp.presenceeventpackage.document.PresenceDocumentFactoryImpl`.
EscMaxDocumentSize	The maximum size, in bytes, for the contents of a publication. If a client attempts to publish a document that is larger than the specified size, the server sends the 413 response, *Request entity too large*. The default value is 10000.
ESCMaxExpires	The maximum time, in seconds, for a publication to expire. The default value is 3600.
ESCMaxPubPerRes	The maximum number of publications allowed per resource. If the maximum number has been reached for a resource when a new publish is received, the server sends the 503 Response (*Service Unavailable*).
ESCMinExpires	The minimum time, in seconds, for a publication to expire. The default is 60.
EventStateCompositor	The class name of the `EventStateCompositor`. The default value is `oracle.sdp.presenceeventpackage.PublishControl`.
Name	The name of this event package. The default value is *Presence*.
Notifier	The name of the `Notifier` class. The default value is `oracle.sdp.presenceeventpackage.PresenceSubscriptionControl`.
NotifierMaxDocumentSize	The maximum size for a SUBSCRIBE request.
NotifierMaxExpires	The maximum time, in seconds, for a SUBSCRIBE to expire. The default is 3600.

Table 9–4 (Cont.) Attributes of the PresenceEventPackage

Attribute	Value/Description
NotifierMaxNoOfSubsPerRes	The maximum number of subscriptions allowed per resource. If the maximum number has been reached for a resource, then a new presence subscribe is received and the server sends the 503 Response (*Service Unavailable*).
NotifierMinExpires	The minimum time, in seconds, for a SUBSCRIBE to expire.
ResourceManagerClassName	The name of the ResourceManager class. The default is `oracle.sdp.presenceeventpackage.PresentityManagerImpl`.

9.2.6 PresenceWInfoEventPackage

As described in RFC 3857, a Watcher Information Event Package monitors the resources in another event package to ascertain the state of all of the subscriptions to that resource. This information is then sent to the subscribers of the Watcher Information Event Package. As a result, the watcher learns of changes in the monitored resources subscriptions.

The PresenceWInfoEventPackage MBean (described in Table 9–5) sets the subscription state information for the Watcher Information Event Package.

Table 9–5 Attributes of the WatcherinfoEventPackage

Attribute	Description/Value
Description	A description of the PresenceWInfoEventPackage. For example: *The event package that enables watcherinfo*.
DocumentFactory	The name of the `DocumentFactory` class. The default is `oracle.sdp.eventnotificationservice.DocumentFactoryImpl`.
Name	The name of the event package. The default value is *presence.winfo*.
Notifier	The `Notifier` class name. The default value is `oracle.sdp.presenceeventpackage.PresenceSubscriptionControl`.
NotifierMaxDocumentSize	The maximum document size for SUBSCRIBE request.
NotifierMaxExpires	The maximum time, in seconds, for a SUBSCRIBE to expire. The default is 3600.
NotifierMaxNoSubsPerRes	The maximum number of subscriptions allowed per resource. If the maximum number has been reached for a resource when a new presence subscribe is received, the server will send a 503 Response (*Service Unavailable*). The default value is 100.
NotifierMinExpires	The minimum time, in seconds, for a SUBSCRIBE to expire.
ResourceManagerClassName	The name of the `ResourceManager` class. The default is `oracle.sdp.winfoeventpackage.WatcherinfoResourceManager`.

9.2.7 UA-ProfileEventPackage

Table 9–6 describes the attributes of the UA-ProfileEventPackage MBean.

Table 9–6 Attributes of the UA-Profile Event Package

Attributes	Description/Value
Description	A description of the UA-ProfileEventPackage. The default value is *The event package that enables the ua-profile*.
Document Factory	The `Document Factory` class name. The default value is: `oracle.sdp.eventnotificationservice.DocumentFactoryImpl`
Name	The name of the event package. The default value is *ua-profile*.

Table 9–6 (Cont.) Attributes of the UA-Profile Event Package

Attributes	Description/Value
Notifier	The name of the `Notifier` class. The default value is: `oracle.sdp.presenceeventpackage.PresenceSubscriptionControl`
NotifierMaxDocumentSize	The maximum document size for a SUBSCRIBE request.
NotifierMaxExpires	The maximum time, in seconds, for a SUBSCRIBE to expire. The default is 6000.
NotifierMaxNoOfSubsPerRes	The maximum number of subscriptions allowed per resource. If the maximum number has been reached for a resource when a new presence subscribe is received, the server will send a 503 Response (*Service Unavailable*). The default value is 100.
NotifierMinExpires	The minimum time, in seconds, for a SUBSCRIBE to expire. The default value is 60.
ResourceManager	The name of the Resource Manager class. The default value is: `oracle.sdp.winfoeventpackage.WatcherinfoResourceManager`

9.2.8 Command Service (XDMS Provisioning)

The Command Service MBean enables user provisioning of the XDMS.

9.2.9 XCapConfigManager

The XCapConfigManager MBean controls the configuration of the XDMS, the repository of the XML documents containing user presence rules and hard state information. The XCapConfigManager MBean settings can be ignored if the XDMS is external to the Presence Server.

Table 9–7 Attributes of the XCapConfigManager MBean

Attribute Name	Description/Value
CreateNonExistingUserstore	Set to *true* to create a user store if one does not exist when storing a document; otherwise, set to *false*. If the parameter is set to *false* and a client tries to store a document for a user that does not exist, then the store fails. If the parameter is set to *true*, then the user will first be created in the XDMS and then the document will be stored. The default value is *true*.
MaxContentLength	The maximum size, in bytes, for an XCAP request. The default is 1MB. You can increase or decrease the size of the document. If you increase the document size, then you must be sure to that there is sufficient disk space to accommodate the XDMS document * the number of users * the number of applications. If you set a smaller per-document size, then this calculation is reduced to the sum of (`max_doc_size_n` * number of users) where each `max_doc_size_n` is specific to application `n`. The default size for the resource-lists document is also 1 MB.
PersistenceRootUrl	The storage location for the XDMS documents. The default is *jpa:xdms*, which means documents will be persisted into an Oracle database through the JDBC connector configured during installation. To change the persistence location to filesystem, use a file url (for example, a value of file:`/tmp/var/xcaproot` will set the storage location to be on the local filesystem in the directory `/tmp/var/xcaproot`). If you choose a local filesystem, make sure the directory is writable by the user running the Presence Server.
PidfManipulationAuid	The ID of the application usage for PIDF (Presence Information Data Format) manipulation. The default value is *pidf-manipulation*.

Table 9–7 (Cont.) Attributes of the XCapConfigManager MBean

Attribute Name	Description/Value
PidfManipulationDocname	The document name for pidf manipulation application usage. For example: *hardstate*. The default value is *hardstate*.
PresRulesAU	The name of the pres-rules application usage. The default value is *pres-rules*.
PresRulesDocName	The name of the pres-rules document. The default value is *presrules*.
PublicContentServerRootUrl	The URL to the public content server root. The URL must be set to the public URL of the content server (that is, the URL of the authentication HTTP proxy server).
PublicXCAPRootUrl	The URL to the public XDMS root, entered as *http://<your.xdms.domain.com>/services/*. For example, enter *http://127.0.0.1:8001/services.*
RequireAssertedIdentity	Set to *true* if all HTTP/XDMS requests require an asserted identity header; otherwise, set this parameter to *false*. Setting this attribute to true requires all XCAP traffic to be authenticated by the Section 9.2.10, "Aggregation Proxy". If this attribute is set to *true*, then any incoming XCAP request that lacks an asserted identity is denied access.

9.2.10 Aggregation Proxy

The Aggregation Proxy is a server-side entry point for XCAP clients and Web Service calls and will authenticate incoming traffic by providing identify assertion. This component acts as the gatekeeper for the trusted domain that houses the Presence Server and the XDMS.

The attributes of the Aggregation Proxy MBean (Table 9–8) enable you to set the type of identity assertion that is appropriate to the XDMS. In addition, you set the host and port of the Web Server and XDMS that receive the proxied traffic from the Aggregation Proxy.

Table 9–8 Attributes of the Aggregation Proxy

Attribute	Description
AssertedIdentityType	Enter the number corresponding to the identity header inserted into proxied HTTP requests that is appropriate to the XDMS: 1. X_3GPP_ASSERTED_IDENTITY (the default) 2. X_3GPP_INTENDED_IDENTITY 3. X_XCAP_ASSERTED_IDENTITY
ContentHost	host name of the Content Server where the Aggregation Proxy sends proxied requests.
ContentPort	The port number of the Content Server where the Aggregation Proxy sends proxied requests.
ContentRoot	The root URL of the Content Server.
IgnoreUserpartCase	Set to *true* if case-sensitive handling of the user name is not required.
Realm	Realm (for ex: example.com) that is used to create the sip/sips address that's inserted in the P-Asserted-Identity header.
TrustedHosts	A comma-separated list of IP addresses of trusted hosts. Asserted identity headers are removed from requests with addresses that are not included in this list.

Table 9–8 (Cont.) Attributes of the Aggregation Proxy

Attribute	Description
XCAPHost	The host name of the XDMS to which the Aggregation Proxy proxies requests.
XCAPPort	The port of the XDMS to which the Aggregation Proxy proxies requests.
XCAPRoot	The root URL of the XDMS.

9.2.11 Configuring Default Application Router for OPTIONS

Follow these steps to configure DAR for handling OPTIONS requests:

1. Open console application in your browser (for example: http://owlcs.example.com:7001/console).
2. Enter Username and Password to log in.
3. In the left pane, click Sip Server.
4. Click the Application Router tab.
5. Under AR configuration data, depending on whether you want to route the OPTIONS request through proxyregistrar, enter:

   ```
   OPTIONS: ("proxyregistrar", "DAR:From", "TERMINATING", "", "NO_ROUTE", "0")
   ```

   OR if you want the request to go through OPTIONSResponder application, enter:

   ```
   OPTIONS : ("optionsresponder", "DAR:From", "TERMINATING", "", "NO_ROUTE",
   "0")
   ```

Note: Options Responder application should have been chosen while extending the domain if you choose the latter DAR configuration.

9.3 Configuring Presence Web Services

OWLCS enables Web Service clients to access presence services through its support of the Parlay X Presence Web Service as defined in *Open Service Access, Parlay X Web Services, Part 14, Presence ETSI ES 202 391-14*. A Parlay X Web Service enables an HTTP Web Service client to access such presence services as publishing and subscribing to presence information. The Parlay X Presence Web Service does not require developers to be familiar with the SIP protocol to build such a Web-based client; instead, Parlay X enables Web developers to build this client using their knowledge of Web Services.

The Presence Web Services application contains the following MBeans that enable you to configure a Web Services deployment server:

- Section 9.3.1, "PresenceSupplierWebService"
- Section 9.3.2, "PresenceConsumerWebService"
- Section 9.3.3, "MessagingWebServiceConfig"

The above MBeans contain attributes for managing presence publication and watcher subscriptions enabled through the OWLCS implementation of Presence Consumer and Presence Supplier interfaces.

9.3.1 PresenceSupplierWebService

The PresenceSupplierWebService MBean (described in Table 9–9) enables you to manage the presence data published to watchers. See Section 9.2.10, "Aggregation Proxy" for more information about aggregation proxy mbean configuration.

Table 9–9 Attributes of the PresenceSupplierWebService MBean

Attributes	Description
Expires	The default expiry time, in seconds, for the PUBLISH of a presence status. The value entered for this attribute should be optimized to match that entered for the *SessionTimeout* attribute.
PIDFManipulationAU	The name of the application usage for PIDF (Presence Information Data Format) manipulation. The default value is *pidf-manipulation*.
PidfManipulationDocname	The document name for pidf manipulation application usage. For example: *hardstate*. If the URI contains a domain name instead of an IP address, then you must configure the DNS Server. The default value is *hardstate*. Note that this value must match the value you entered when configuring XDMS.
PresRulesAU	The name of the pres-rules application usage. The default value is *pres-rules*.
PresRulesDocname	The name of the pres-rules document. The default value is *presrules*.
PublicXCAPRootUrl	The URL to the public XDMS root, entered as *http://<your.xdms:domain.com>/services/*. For example, enter *http://127.0.0.1:8001/services.*
SessionTimeout	The timeout of the HTTP session, in seconds. The value entered for this attribute should be optimized to match the value entered for the *Expires* attribute. This timeout takes effect for new sessions only.
SIPOutboundProxy	The IP address of the outbound proxy server where all requests are sent on the first hop. Enter this address in the following format: `sip:<IP address>[:port];lr;transport=TCP` You can also enter the default port (5060) in this address. For example, enter *sip:127.0.0.1:5060;lr;transport=TCP*. If you do not define this attribute, then no outbound proxy will be used.

9.3.2 PresenceConsumerWebService

The PresenceConsumerWebService MBean (described in Table 9–10) enables you to set the duration of watcher subscriptions.

Table 9–10 Attributes of the PresenceConsumerWebService MBean

Attribute	Value
Expires	The default expiry time, in seconds, for watcher subscriptions. The value entered for this attribute should be optimized to match the value entered for the *SessionTimeout* attribute.
SessionTimeout	The timeout of the HTTP session, in seconds. The value entered for this attribute should be optimized to match the value entered for the *Expires* attribute. This timeout takes effect for new sessions only.

Table 9–10 (Cont.) Attributes of the PresenceConsumerWebService MBean

Attribute	Value
SIPOutboundProxy	The IP address of the outbound proxy server where all requests are sent on the first hop. Enter this address in the following format: `sip:<IP address>[:port];lr;transport=TCP` You can also enter the default port (5060) in this address. For example, enter *sip:127.0.0.1:5060;lr;transport=TCP*. If you do not define this attribute, then no outbound proxy will be used.

9.3.3 MessagingWebServiceConfig

MessagingWebServiceConfig (described in Table 9–11) enables you to designate how and when to delete messages stored by the web service.

Table 9–11 MessagingWebServiceConfig attributes

Attribute	Value
SessionTimeout	The timeout of the HTTP session, in seconds. The value entered for this attribute should be optimized to match the value entered for the *Expires* attribute. This timeout takes effect for new sessions only
SIPOutboundProxy	The IP address of the outbound proxy server where all requests are sent on the first hop. Enter this address in the following format: `sip:<IP address>[:port];lr;transport=TCP` If you do not define this attribute, then no outbound proxy will be used.
MessageLifetime	Set the time in seconds after which messages expire from the message store. Setting this to 0 will cause messages to be kept in the store indefinitely (never expire). Messages will stay in the message store for at a maximum MessageLifetime + MessageScanPeriod seconds. Setting this attribute has immediate effect (for instance, reducing the value could cause some messages to be immediately expired if they are older than the lifetime).
MessageScanPeriod	Set the period in seconds for scanning for and deleting expired messages. Setting this to 0 disables scanning. Setting this attribute has immediate effect.

Part III

Configuring Diameter

This Part contains the following chapter:

10

Configuring Diameter Client Nodes and Relay Agents

The following sections describe how to configure individual servers to act as Diameter nodes or relays in a Oracle WebLogic Communication Services domain:

10.1 Overview of Diameter Protocol Configuration

A typical Oracle WebLogic Communication Services domain includes support for the Diameter base protocol and one or more IMS Diameter interface applications (Sh, Ro, Rf) deployed to engine tier servers that act as Diameter client nodes. SIP Servlets deployed on the engines can use the available Diameter applications to initiate requests for user profile data, accounting, and credit control, or to subscribe to and receive notification of profile data changes.

One or more server instances may be also be configured as Diameter relay agents, which route Diameter messages from the client nodes to a configured Home Subscriber Server (HSS) or other nodes in the network, but do not modify the messages. Oracle recommends configuring one or more servers to act as relay agents in a domain. The relays simplify the configuration of Diameter client nodes, and reduce the number of network connections to the HSS. Using at least two relays ensures that a route can be established to an HSS even if one relay agent fails.

Note: In order to support multiple HSSs, the 3GPP defines the Dh interface to look up the correct HSS. Oracle WebLogic Communication Services does not provide a Dh interface application, and can be configured only with a single HSS.

Note that relay agent servers do not function as either engine or SIP data tier instances—they should not host applications, store call state data, maintain SIP timers, or even use SIP protocol network resources (sip or sips network channels).

Oracle WebLogic Communication Services also provides simulator applications for the Sh and Ro protocols. You can use the simulator applications for testing while developing Sh and Ro clients. The simulator applications are not intended for deployment to a production system.

10.2 Steps for Configuring Diameter Client Nodes and Relay Agents

To configure Diameter support in a Oracle WebLogic Communication Services domain, follow these steps:

1. Install the Oracle WebLogic Communication Services Diameter Domain. Install the Diameter domain, which contains a sample configuration and template applications configured for different Diameter node types. You may use the Diameter domain as a template for your own domain, or to better understand how different Diameter node types are configured.
2. Enable the Diameter console extension. If you are working with the sample Diameter domain, the Diameter console extension is already enabled. If you are starting with a basic Oracle WebLogic Communication Services domain, edit the `config.xml` file to enable the extension.
3. Create Diameter network channels. Create the network channels necessary to support Diameter over TCP, TLS, or SCTP transports on engine tier servers and relays.
4. Create and configure the Diameter nodes. Configure the Diameter protocol client applications on engine tier servers with the host name, peers, and routes to relay agents or other network elements, such as an HSS. You can also configure Diameter nodes that operate in standalone mode, without a Oracle WebLogic Communication Services instance.

The sections that follow describe each step in detail. See also the Section 10.7, "Example Domain Configuration".

10.3 Installing the Diameter Domain

The Configuration Wizard includes a Diameter domain template that creates a domain having four Oracle WebLogic Communication Services instances:

- An Administration Server (AdminServer)
- A Diameter Sh client node (hssclient)
- A Diameter relay node (relay)
- An HSS simulator (hss)

You can use the installed Diameter domain as the basis for creating your own domain. Or, you can use the customized Diameter Web Applications as templates for configuring existing Oracle WebLogic Communication Services instances to function as HSS client or relay agent nodes. The configuration instructions in the sections that follow assume that you have access to the Diameter domain configuration. Follow these steps to install the domain:

1. Change to the `WLS_HOME\common\bin` directory, where `WLS_HOME` is the directory in which you installed the Oracle WebLogic Server 10*g* Release 3 portion

Oracle WebLogic Communication Services (for example, `c:\bea\wlserver_10.3\common\bin`).

2. Execute the `config.cmd` or `config.sh` script to launch the Configuration Wizard.
3. Select Create a new WebLogic Domain and click **Next**.
4. Select Base this domain on an existing template, and click **Browse**.
5. Select the `diameterdomain.jar` template and click **OK**.
6. Click **Next**.
7. Enter a username and password for the Administrator of the new domain, and click **Next**.
8. Select the startup mode and JDKs to use with the new domain, and click **Next**.
9. Select No to accept the default template options, and click **Next**.
10. Click Create to create the new domain using the default domain name and domain location (`/user_projects/domains/diameter`).
11. Click Done.

Table 10–3 describes the server configuration installed with the Diameter domain.

Table 10–1 Key Configuration Elements of the Diameter Domain

Server Name	Network Channel Configuration	Diameter Applications	Notes
`AdminServer`	n/a	n/a	The Administration Server provides no SIP or Diameter protocol functionality.
`hssclient`	diameter (TCP over port 3868) sip (UDP/TCP over port 5060)	`WlssShApplication`	The `hssclient` engine functions as a Diameter Sh client node. The server contains network channels supporting both SIP and Diameter traffic. The Diameter node configuration deploys `WlssShApplication` (`com.bea.wcp.diameter.sh.WlssShApplication`) to provide IMS Sh interface functionality for deployed SIP Servlets.
`relay`	diameter (TCP over port 3869)	`RelayApplication`	The `relay` engine functions as a Diameter Sh relay node. The server contains a network channel to support both Diameter traffic. The server does not contain a channel to support SIP traffic, as a relay performs no SIP message processing. The Diameter node configuration deploys `RelayApplication` (`com.bea.wcp.diameter.relay.RelayApplication`) to provide relay services. The node configuration also defines a realm-based route for relaying messages from the `hssclient` engine.
`hss`	diameter (TCP over port 3870)	`HssSimulator`	The `hss` engine's Diameter node configuration deploys only the `HssSimulator` application (`com.bea.wcp.diameter.sh.HssSimulator`). The server is configured with a Diameter network channel.

10.4 Enabling the Diameter Console Extension

Oracle WebLogic Communication Services provides a console extension to help you create and configure Diameter nodes. The actual configuration generated by the extension is stored in a `diameter.xml` configuration file, stored in the `config/custom` subdirectory of the domain directory.

The sample Diameter domain already enables the Diameter console extension. If you are working with a domain that does not enable the extension, edit the `config.xml` file for the domain to specify the custom resource for the extension. Example 10–1 highlights the `config.xml` entries necessary to enable the console extension. Use a text editor to add the highlighted lines in the correct location in `config.xml`.

Example 10–1 config.xml Entries for Enabling the Diameter Console Extension

```
<custom-resource>
    <name>sipserver</name>
    <target>hssclient</target>
    <descriptor-file-name>custom/sipserver.xml</descriptor-file-name>

<resource-class>com.bea.wcp.sip.management.descriptor.resource.SipServerResource</
resource-class>

<descriptor-bean-class>com.bea.wcp.sip.management.descriptor.beans.SipServerBean</
descriptor-bean-class>
  </custom-resource>
  <custom-resource>
    <name>diameter</name>
    <target>hssclient,relay,hss</target>
    <deployment-order>200</deployment-order>
    <descriptor-file-name>custom/diameter.xml</descriptor-file-name>
    <resource-class>com.bea.wcp.diameter.DiameterResource</resource-class>

<descriptor-bean-class>com.bea.wcp.diameter.management.descriptor.beans.DiameterBe
an</descriptor-bean-class>
  </custom-resource>
  <custom-resource>
    <name>ProfileService</name>
    <target>hssclient</target>
    <deployment-order>300</deployment-order>
    <descriptor-file-name>custom/profile.xml</descriptor-file-name>

<resource-class>com.bea.wcp.profile.descriptor.resource.ProfileServiceResource</re
source-class>

<descriptor-bean-class>com.bea.wcp.profile.descriptor.beans.ProfileServiceBean</de
scriptor-bean-class>
  </custom-resource>
  <admin-server-name>AdminServer</admin-server-name>
</domain>
```

10.5 Creating TCP, TLS, and SCTP Network Channels for the Diameter Protocol

The Oracle WebLogic Communication Services Diameter implementation supports the Diameter protocol over the TCP, TLS, and SCTP transport protocols. (SCTP transport is provided with certain restrictions as described in Section 10.5.2, "Configuring and Using SCTP for Diameter Messaging".)

To enable incoming Diameter connections on a server, you must configure a dedicated network channel of the appropriate protocol type:

- "diameter" channels use TCP transport
- "diameters" channels use TCP/TLS transport
- "diameter-sctp" channels use TCP/SCTP transport.

Servers that use a TCP/TLS channel for Diameter (diameters channels) must also enable two-way SSL. Oracle WebLogic Communication Services may automatically upgrade Diameter TCP connections to use TLS as described in the Diameter specification (RFC 3558).

To configure a TCP or TCP/TLS channel for use with the Diameter provider, follow these steps:

1. Access the Administration Console for the Oracle WebLogic Communication Services domain.
2. Click **Lock & Edit** to obtain a configuration lock.
3. In the left pane, select the name of the server to configure.
4. In the right pane, select **Protocols > Channels** to display the configured channels.
5. Click **New** to configure a new channel.
6. Fill in the fields of the Identity Properties page as follows:
   - **Name:** Enter an administrative name for this channel, such as "Diameter TCP/TLS Channel."
   - **Protocol:** Select "diameter" to support the TCP transport, "diameters" to support both TCP and TLS transports, or "diameter-sctp" to support TCP transport.

     Note: If a server configures at least one TLS channel, the server operates in TLS mode and will reject peer connections from nodes that do not support TLS (as indicated in their capabilities exchange).
7. Click **Next** to continue.
8. Fill in the fields of the Network Channel Addressing page as follows:
   - **Listen Address:** Enter the IP address or DNS name for this channel. On a multi-homed machine, enter the exact IP address of the interface you want to configure, or a DNS name that maps to the exact IP address.
   - **Listen Port:** Enter the port number used to communication via this channel. Diameter nodes conventionally use port 3868 for incoming connections.
   - **External Listen Port**: Re-enter the Listen Port value.
9. Click **Next** to continue.
10. Chose attributes in the Network Channel Properties page as follows:
    - **Enabled**: Select this attribute to ensure that the new channel accepts network traffic.
    - **Tunneling Enabled**: Un-check this attribute for Diameter channels.
    - **HTTP Enabled for this Protocol**: Un-check this attribute for Diameter channels.

- **Outbound Enabled**: Select this attribute to ensure that the node can initiate Diameter messages using the channel.

11. Click **Next** to continue.
12. For "diameters" channels, select the following two attributes:
    - **Two Way SSL Enabled**: Two-way SSL is required for TLS transport.
    - **Client Certificate Enforced**: Select this attribute to honor available client certificates for secure communication.
13. Click **Finish** to create the new channel.
14. Select the name of the newly-created channel in the Network Channel table.
15. Display the advanced configuration items for the newly-created channel by clicking the Advanced link.
16. Change the **Idle Connection Timeout** value from the default (65 seconds) to a larger value that will ensure the Diameter connection remains consistently available.

    Note: If you do not change the default value, the Diameter connection will be dropped and recreated every 65 seconds with idle traffic.

17. Click **Save**.
18. Click **Activate Changes**.

The servers installed with the Diameter domain template include network channel configurations for Diameter over TCP transport. Note that the relays server includes only a diameter channel and *not* a sip or sips channel. Relay agents should not host SIP Servlets or other applications, therefore no SIP transports should be configured on relay server nodes.

10.5.1 Configuring Two-Way SSL for Diameter TLS Channels

Diameter channels that use TLS (diameters channels) require that you enable two-way SSL, which is disabled by default. Select Two Way Enabled SSL in the steps above to enable two-way SSL.

10.5.2 Configuring and Using SCTP for Diameter Messaging

SCTP is a reliable, message-based transport protocol that is designed for use in telephony networks. SCTP provides several benefits over TCP:

- SCTP preserves the internal structure of messages when transmitting data to an endpoint, whereas TCP transmits raw bytes that must be received in order.
- SCTP supports multihoming, where each endpoint may have multiple IP addresses. The SCTP protocol can transparently failover to another IP address should a connection fail.
- SCTP provides multistreaming capabilities, where multiple streams in a connection transmit data independently of one another.

Oracle WebLogic Communication Services supports SCTP for Diameter network traffic, with several limitations:

- Only 1 stream per connection is currently supported.

- SCTP can be used only for Diameter network traffic; SIP traffic cannot use a configured SCTP channel.
- TLS is not supported over SCTP.

In addition, Oracle WebLogic Communication Services only supports SCTP channels on the following software platforms:

- Red Hat Enterprise Linux 4.0 AS, ES, WS (Kernel 2.6.9, GCC 3.4 or higher) on 32- or 64-bit hardware
- Sun Solaris 10 on SPARC

SCTP channels can operate on either IPv4 or IPv6 networks. Section 10.5 describes how to create a new SCTP channel. To enable multihoming capabilities for an existing SCTP channel, specify the IPv4 address `0.0.0.0` as the listen address for the channel (or use the `::` address for IPv6 networks).

10.6 Configuring Diameter Nodes

The Diameter node configuration for Oracle WebLogic Communication Services engines is stored in the `diameter.xml` configuration file (`domain_home/config/custom/diameter.xml`). If you want to provide diameter services (client, server, or relay functionality) on an engine tier server, you must create a new node configuration and target the configuration to an existing engine server instance.

Diameter node configurations are divided into several categories:

- General configuration defines the host identity and realm for the node, as well as basic connection information and default routing behavior.
- Application configuration defines the Diameter application(s) that run on the node, as well as any optional configuration parameters passed to those applications.
- Peer configuration defines the other Diameter nodes with which this node operates.
- Routes configuration defines realm-based routes that the node can use when resolving messages.

The sections that follow describe how to configure each aspect of a Diameter node.

10.6.1 Creating a New Node Configuration (General Node Configuration)

Follow these steps to create a new Diameter node configuration and target it to an existing Oracle WebLogic Communication Services engine tier instance:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Click Lock & Edit to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Click New in the right pane to create a new Diameter configuration.
5. Fill in the fields of the Create a New Configuration page as described in Table 10–3, then click Finish.

Table 10–2 Diameter Node General Configuration Properties

Property Name	Description
Name	Enter the an administrative name for this Diameter node configuration.
Host	Enter the host identity of this Diameter node, or leave the field blank to automatically assign the host name of the target engine tier server as the Diameter node's host identity. Note that the host identity may or may not match the DNS name. When configuring Diameter support for multiple Sh client nodes, it is best to omit the `host` element from the `diameter.xml` file. This enables you to deploy the same Diameter Web Application to all servers in the engine tier cluster, and the host name is dynamically obtained for each server instance.
Realm	Enter the realm name for which this node has responsibility, or leave the field blank to use the domain name portion of the target engine tier server's fully-qualified host name (for example, host@oracle.com). You can run multiple Diameter nodes on a single host using different realms and listen port numbers. **Note:** An HSS, Application Server, and relay agents must all agree on a realm name or names. The realm name for the HSS and Application Server need not match.
Address	Enter the listen address for this Diameter node, using either the DNS name or IP address, or leave the field blank to use the host identity as the listen address. **Note:** The host identity may or may not match the DNS name of the Diameter node. Oracle recommends configuring the Address property with an explicit DNS name or IP address to avoid configuration errors.
TLS	Select this option if the Diameter node us configured with support for TLS (diameters network channels). This field is used to advertise TLS capabilities when the node is interrogated by another Diameter node.
Debug	Select this option if you want to enable debug message output. Debug messages are disabled by default.
Message Debug	Select this option if you want to enable tracing for Diameter messages processed by this node. Message tracing is disabled by default.
Dynamic Peers Allowed	Select this option to allow dynamic discovery of Diameter peer nodes. Dynamic peer support is disabled by default. Oracle recommends enabling dynamic peers only when using the TLS transport, because no access control mechanism is available to restrict hosts from becoming peers.
Peer Retry Delay	Enter the amount of time, in seconds, this node waits before retrying a request to a Diameter peer. The default value is 30 seconds.
Request Timeout	Enter the amount of time, in milliseconds, this node waits for an answer message before timing out.
Watchdog Timeout	Enter the number of seconds this node uses for the value of the Diameter Tw watchdog timer interval.
Targets	Enter one or more target engine tier server names. The Diameter node configuration only applies to servers listed in this field.

Table 10–2 (Cont.) Diameter Node General Configuration Properties

Property Name	Description
Default Route Action	Specify an action type that describes the role of this Diameter node when using a default route. The value of this element can be one of the following: ■ none ■ local ■ relay ■ proxy ■ redirect
Default Route Servers	Specifies one or more target servers for the default route. Any server you include in this element must also be defined as a peer to this Diameter node, or dynamic peer support must be enabled.

6. Click **Activate Changes** to apply the configuration to target servers.

After creating a general node configuration, the configuration name appears in the list of Diameter nodes. You can select the node to configure Diameter applications, peers, and routes, as described in the sections that follow.

10.6.2 Configuring Diameter Applications

Each Diameter node can deploy one or more applications. You configure Diameter applications in the Administration Console using the **Configuration > Applications** page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Click **Lock & Edit** to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the **Configuration > Applications** tab.
6. Click **New** to configure a new Diameter application, or select an existing application configuration from the table.
7. Fill in the application properties as follows:
   - **Application Name**: Enter a name for the application configuration.
   - **Class Name**: Enter the classname of the application to deploy on this node.
   - **Parameters**: Enter optional parameters to pass to the application upon startup.
8. Click **Finish** to create the new application configuration.
9. Click **Activate Changes** to apply the configuration to the Diameter node.

Oracle WebLogic Communication Services includes several Diameter applications to support clients using the Sh, Rf, and Ro interfaces, Diameter relays, and simulators for the Sh and Ro interfaces. The sections that follow provide more information about configuring these Oracle WebLogic Communication Services Diameter applications.

You can also use the base Diameter API included in Oracle WebLogic Communication Services to create and deploy your own Diameter applications.

10.6.2.1 Configuring the Sh Client Application

The Sh client application is implemented as a provider to the Profile Service API. The application transparently generates and responds to the Diameter command codes defined in the Sh application specification. The Profile Service API enables SIP Servlets to manage user profile data as an XML document using XML Document Object Model (DOM). Subscriptions and notifications for changed profile data are managed by implementing a profile listener interface in a SIP Servlet.

The Diameter nodes on which you deploy the Sh client application should be configured with:

- The host names of any relay agents configured in the domain, defined as Diameter peer nodes. If no relay agents are used, all engine tier servers must be added to the list of peers, or dynamic peers must be enabled.
- One or more routes to access relay agent nodes (or the HSS) in the domain.

To configure the Sh client application, you specify the `com.bea.wcp.diameter.sh.WlssShApplication` class. `WlssShApplication` accepts the following parameters:

- `destination.host` configures a static route to the specified host. Include a `destination.host` param definition only if servers communicate directly to an HSS (static routing), without using a relay agent. Omit the `destination.host` param completely when routing through relay agents.
- `destination.realm`—configures a static route to the specified realm. Specify the realm name of relay agent servers or the HSS, depending on whether or not the domain uses relay agents.

Example 10–2 shows a sample node configuration for an Sh client node that uses a relay.

Example 10–2 Sample Diameter Node Configuration with Sh Client Application

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
    <name>hssclient</name>
    <target>hssclient</target>
    <host>hssclient</host>
    <realm>oracle.com</realm>
    <!-- Omit the host and realm elements to dynamically assign the host name
     and domain name of individual engine tier servers. - >
    <message-debug-enabled>true</message-debug-enabled>
    <application>
      <name>WlssShApplication</name>
      <class-name>com.bea.wcp.diameter.sh.WlssShApplication</class-name>
      <param>
      <!-- Include a destination.host param definition only if servers will
           communicate directly to an HSS (static routing), without using
           a relay agent. Omit the destination.host param completely when
           routing through relay agents. - >
      <!-- Specify the realm name of relay agent servers or the HSS,
           depending on whether or not the domain uses relay agents. - >
        <name>destination.realm</name>
        <value>hss.com</value>
      </param>
```

```
    </application>
    <peer>
    <!-- Include peer entries for each relay agent server used in the domain.
         If no relay agents are used, include a peer entry for the HSS
         itself, as well as for all other Sh client nodes (all other engine
         tier servers in the domain).
         Alternately, use the allow-dynamic-peers functionality in
         combination with TLS transport to allow peers to be recognized
         automatically. - >
      <host>relay</host>
      <address>localhost</address>
      <!-- The address element can specify either a DNS name or IP address,
           whereas the host element must specify a diameter host identity.
           The diameter host identity may or may not match the DNS name. - >
      <port>3869</port>
    </peer>
    <!-- Enter a default route to a selected relay agent. If the domain does
         not use a relay agent, specify a default route to relay messages
         directly to the HSS. - >
    <default-route>
      <action>relay</action>
      <server>relay</server>
    </default-route>
  </configuration>
</diameter>
```

10.6.2.2 Configuring the Rf Client Application

The Oracle WebLogic Communication Services Rf client application enables SIP Servlets to issue offline charging messages using the IMS Rf interface. To configure the Rf application, specify the class `com.bea.wcp.diameter.charging.RfApplication`. The Rf application accepts the following parameters:

- `cdf.host` specifies the host name of the Charging Data Function (CDF).
- `cdf.realm` specifies the realm of the CDF.

10.6.2.3 Configuring the Ro Client Application

The Oracle WebLogic Communication Services Ro client application enables SIP Servlets to issue online charging messages using the IMS Ro interface. To configure the Rf application, specify the class `com.bea.wcp.diameter.charging.RoApplication.` The Ro application accepts the following parameters:

- `ocs.host` specifies the host identity of the Online Charging Function (OCF). The OCF you specify host must also be configured as the peer for the Diameter node on which the Ro application is deployed.
- `ocs.realm` can be used instead of `ocs.host` for realm-based routing when using more than one OCF host. The corresponding realm definition must also exist in the Diameter node's configuration.

10.6.2.4 Configuring a Diameter Relay Agent

Relay agents are not required in a Diameter configuration, but Oracle recommends using at least two relay agent servers to limit the number of direct connections to the HSS, and to provide multiple routes to the HSS in the event of a failure.

Note: You must ensure that relay servers *do not* also act as Oracle WebLogic Communication Services engine tier servers or SIP data tier servers. This means that the servers should not be configured with "sip" or "sips" network channels.

Relay agent nodes route Sh messages between client nodes and the HSS, but they do not modify the messages except as defined in the Diameter Sh specification. Relays always route responses from the HSS back the client node that initiated the message, or the message the response is dropped if that node is unavailable.

To configure a Diameter relay agent, simply configure the node to deploy an application with the class `com.bea.wcp.diameter.relay.RelayApplication`.

The node on which you deploy the relay application should also configure:

- All other nodes as peers to the relay node.
- A default route that specifies the relay action.

Example 10–3 shows the sample `diameter.xml` configuration for a relay agent node.

Example 10–3 Diameter Relay Node Configuration

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
<name>relay</name>
    <target>relay</target>
    <host>relay</host>
    <realm>oracle.com</realm>
    <message-debug-enabled>true</message-debug-enabled>
    <application>
      <name>RelayApplication</name>
      <class-name>com.bea.wcp.diameter.relay.RelayApplication</class-name>
    </application>
    <!-- Define peer connection information for each Diameter node, or use
         the allow-dynamic-peers functionality in combination with TLS
         transport to allow peers to be recognized automatically. - >
    <peer>
      <host>hssclient</host>
      <address>localhost</address>
      <port>3868</port>
    </peer>
    <peer>
      <host>hss</host>
      <address>localhost</address>
      <port>3870</port>
    </peer>
    <route>
      <realm>oracle.com</realm>
      <application-id>16777217</application-id>
      <action>relay</action>
      <server>hssclient</server>
    </route>
    <!-- Enter a default route for this agent to relay messages
         to the HSS. - >
    <default-route>
```

```
      <action>relay</action>
      <server>hss</server>
    </default-route>
  </configuration>
</diameter>
```

10.6.2.5 Configuring the Sh and Rf Simulator Applications

Oracle WebLogic Communication Services contains two simulator applications that you can use in development or testing environments to evaluate Diameter client applications. To configure a simulator application, you simply deploy the corresponding class to a configured Diameter node:

- `com.bea.wcp.diameter.sh.HssSimulator` simulates an HSS in your domain for testing Sh client applications.
- `com.bea.wcp.diameter.rf.RfSimulator` simulates an CDF host for testing Rf client applications

> **Note:** These simulators are provided for testing or development purposes only, and is not meant as a substitute for a production HSS or CDF.

Diameter nodes that deploy simulator applications can be targeted to running engine tier servers, or they may be started as standalone Diameter nodes. When started in standalone mode, simulator applications accept the command-line options described in Table 10–3.

Table 10–3 Command-Line Options for Simulator Applications

Option	Description
`-r, -realm` *realm_name*	Specifies the realm name of the Diameter node.
`-h, -host host_name`	Specifies the host identity of the node.
`-a, -address address`	Specifies the listen address for this node.
`-p, -port port_number`	Specifies the listen port number for this node.
`-d, -debug`	Enables debug output.
`-m, -mdebug`	Enables Diameter message tracing.

10.6.2.6 Enabling Profile Service (using an Sh backend)

As noted earlier, Sh, Ro, and Rf applications can be configured and used separately, but Sh can take advantage of the Profile Service API. To do so:

1. Configure ShApplication in `diameter.xml` (see Example 10–5 for more information).
2. Add a `profile.xml` file to domain_home/config/custom/profile.xml. You can either install the Diameter domain as a template and modify the file, or you can manually create profile.xml as shown in Example 10–4.

Example 10–4 profile.xml sample

```
<profile-service xmlns="http://www.bea.com/ns/wlcp/wlss/profile/300">
  <mapping>
```

```
    <map-by>prefix</map-by>
    <map-by-prefix>
      <provider-prefix-set>
        <name>sh</name>
        <prefix>sh</prefix>
      </provider-prefix-set>
    </map-by-prefix>
  </mapping>
  <provider>
    <name>sh</name>
    <provider-class>com.bea.wcp.profile.ShProviderCached</provider-class>
  </provider>
</profile-service>
```

10.6.3 Configuring Peer Nodes

A Diameter node should define peer connection information for each other Diameter node in the realm, or enable dynamic peers in combination with TLS transport to allow peers to be recognized automatically. You configure Diameter peer nodes in the Administration Console using the Configuration > Peers page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Click **Lock & Edit** to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the **Configuration > Peers** tab.
6. Click **New** to define a new peer entry.
7. Fill in the fields of the Create a New Peer page as follows:
   - **Host**: Enter the peer node's host identity.
   - **Address**: Enter the peer node's address (DNS name or IP address).
   - **Port Number**: Enter the listen port number of the peer node.
   - **Protocol**: Select the protocol used to communicate with the peer (TCP or SCTP).

     Note: Oracle WebLogic Communication Services attempts to connect to the peer using *only* the protocol you specify (TCP or SCTP). The other protocol is not used, even if a connection fails using the selected protocol. TCP is used as by default if you do not specify a protocol.

   - **Watchdog**: Indicate whether the peer supports the Diameter Tw watchdog timer interval.
8. Click **Finish** to create the new peer entry.
9. Click **Activate Changes** to apply the configuration.

10.6.4 Configuring Routes

Certain Diameter nodes, such as relays, should configure realm-based routes for use when resolving Diameter messages. You configure Diameter routes in the

Administration Console using the Configuration > Routes page for a selected Diameter node. Follow these steps:

1. Log in to the Administration Console for the Oracle WebLogic Communication Services domain you want to configure.
2. Click **Lock & Edit** to obtain a configuration lock.
3. Select the Diameter node in the left pane of the Console.
4. Select the name of a Diameter node configuration in the right pane of the Console.
5. Select the **Configuration > Routes** tab.
6. Click **New** to configure a new Route.
7. Fill in the fields of the Create a New Route page as follows:
   - **Name**: Enter an administrative name for the route.
   - **Realm**: Enter the target realm for this route.
   - **Application ID**: Enter the target Diameter application ID for this route.
   - **Action**: Select an action that this node performs when using the configured route. The action type may be one of: none, local, relay, proxy, or redirect.
   - **Server Names**: Enter the names of target servers that will use the route.
8. Click **Finish** to create the new route entry.
9. Click **Activate Changes** to apply the configuration.

See Example 10–3 for an example `diameter.xml` node configuration containing a route entry.

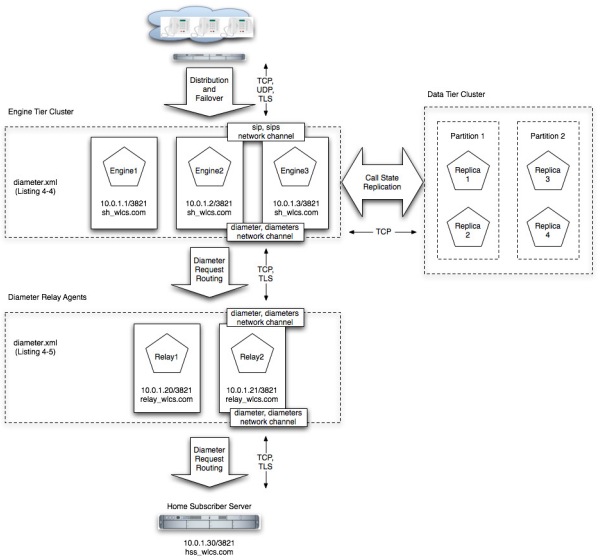
10.7 Example Domain Configuration

This section describes a sample Oracle WebLogic Communication Services configuration that provides basic Diameter Sh protocol capabilities. The layout of the sample domain includes the following:

- Three engine tier servers which host SIP applications and also deploy the Diameter Sh application for accessing user profiles.
- Four SIP data tier servers arranged into two partitions with two replicas each.
- Two servers that act as Diameter relay agents and forward diameter requests to an HSS.

Figure 10–1 shows the individual servers in the sample configuration.

Figure 10–1 Sample Diameter Domain



Example 10–5 shows the contents of the `diameter.xml` file used to configure engine tier servers (Sh Clients) in the sample domain. Example 10–6 shows the `diameter.xml` file used to configure the relay agents.

Example 10–5 diameter.xml Configuration for Sample Engine Tier Cluster (Sh Clients)

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
<configuration>
    <name>clientnodes</name>
    <target>Engine1</target>
    <target>Engine2</target>
    <target>Engine3</target>
    <realm>sh_wlss.com</realm>
    <application>
      <name>WlssShApplication</name>
      <class-name>com.bea.wcp.diameter.sh.WlssShApplication</class-name>
      <param>
        <name>destination.realm</name>
        <value>relay_wlss.com</value>
      </param>
    </application>
    <peer>
      <host>Relay1</host>
      <address>10.0.1.20</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Relay2</host>
      <address>10.0.1.21</address>
      <port>3821</port>
    </peer>
    <default-route>
      <action>relay</action>
      <server>Relay1</server>
```

```
    </default-route>
    <route>
      <action>relay</action>
      <server>Relay2</server>
    </route>
  </configuration>
</diameter>
```

Example 10–6 diameter.xml Configuration for Sample Relay Agents

```
<?xml version='1.0' encoding='utf-8'?>
<diameter xmlns="http://www.bea.com/ns/wlcp/diameter/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls">
  <configuration>
    <name>relaynodes</name>
    <target>Relay1</target>
    <target>Relay2</target>
    <realm>relay_wlss.com</realm>
    <application>
      <name>RelayApplication</name>
      <class-name>com.bea.wcp.diameter.relay.RelayApplication</class-name>
    </application>
    <peer>
      <host>Engine1</host>
      <address>10.0.1.1</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Engine2</host>
      <address>10.0.1.2</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Engine3</host>
      <address>10.0.1.3</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Relay1</host>
      <address>10.0.1.20</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>Relay2</host>
      <address>10.0.1.21</address>
      <port>3821</port>
    </peer>
    <peer>
      <host>hss</host>
      <address>hssserver</address>
      <port>3870</port>
    </peer>
    <default-route>
      <action>relay</action>
      <server>hss</server>
    </default-route>
  </configuration>
</diameter>
```

10.8 Troubleshooting Diameter Configurations

SIP Servlets deployed on Oracle WebLogic Communication Services use the available Diameter applications to initiate requests for user profile data, accounting, and credit control, or to subscribe to and receive notification of profile data changes. If a SIP Servlet performing these requests generates an error similar to:

```
Failed to dispatch Sip message to servlet ServletName
java.lang.IllegalArgumentException: No registered provider for protocol: Protocol
```

The message may indicate that you have not properly configured the associated Diameter application for the protocol. See Section 10.6.2, "Configuring Diameter Applications" for more information.

If you experience problems connecting to a Diameter peer node, verify that you have configured the correct protocol for communicating with the peer in Section 10.6.3, "Configuring Peer Nodes". Keep in mind that Oracle WebLogic Communication Services tries only the protocol you specify for the peer configuration (or TCP if you do not specify a protocol).

Part IV

Oracle User Messaging Services

This Part contains the following chapters

11

Configuring Oracle User Messaging Service

This chapter describes how to configure Oracle User Messaging Service (UMS).

This chapter includes the following topics:

11.1 User Messaging Service Overview

Oracle User Messaging Service enables two-way communication between users and deployed applications. Key features include:

- Support for a variety of messaging channels—Messages can be sent and received through Email, IM (XMPP), SMS (SMPP), and Voice. Messages can also be delivered to a user's SOA/WebCenter Worklist.
- Two-way Messaging—In addition to sending messages from applications to users (referred to as *outbound* messaging), users can initiate messaging interactions (inbound messaging). For example, a user can send an email or text message to a specified address; the message is routed to the appropriate application which can then respond to the user or invoke another process according to its business logic.
- User Messaging Preferences—End users can use a web interface to define preferences for how and when they receive messaging notifications. Applications immediately become more flexible; rather than deciding whether to send to a user's email address or instant messaging client, the application can simply send the message to the user, and let UMS route the message according to the user's preferences.
- Robust Message Delivery—UMS keeps track of delivery status information provided by messaging gateways, and makes this information available to applications so that they can respond to a failed delivery. Or, applications can specify one or more *failover* addresses for a message in case delivery to the initial address fails. Using the failover capability of UMS frees application developers from having to implement complicated retry logic.
- Pervasive integration within Fusion Middleware: UMS is integrated with other Fusion Middleware components providing a single consolidated bi-directional user messaging service.

- Integration with Oracle BPEL—Oracle JDeveloper includes pre-built BPEL activities that enable messaging operations. Developers can add messaging capability to a SOA composite application by dragging and dropping the desired activity into any workflow.
- Integration with Oracle Human Workflow—UMS enables the Human Workflow engine to send actionable messages to and receive replies from users over email.
- Integration with Oracle BAM—Oracle BAM uses UMS to send email alerts in response to monitoring events.
- Integration with Oracle WebCenter—UMS APIs are available to developers building applications for Oracle WebCenter Spaces. The API is a realization of Parlay X Web Services for Multimedia Messaging, version 2.1, a standard web service interface for rich messaging.

11.1.1 Components

There are three types of components that make up Oracle User Messaging Service. These components are standard Java EE applications, making it easy to deploy and manage them using the standard tools provided with Oracle WebLogic Server.

- UMS Server: The UMS Server orchestrates message flows between applications and users. The server routes outbound messages from a client application to the appropriate driver, and routes inbound messages to the correct client application. The server also maintains a repository of previously sent messages in a persistent store, and correlates delivery status information with previously sent messages.
- UMS Drivers: UMS Drivers connect UMS to the messaging gateways, adapting content to the various protocols supported by UMS. Drivers can be deployed or undeployed independently of one another depending on what messaging channels are available in a given installation.
- UMS Client applications: UMS client applications implement the business logic of sending and receiving messages. A UMS client application might be a SOA application that sends messages as one step of a BPEL workflow, or a WebCenter Spaces application that can send messages from a web interface.

In addition to the components that make up UMS itself, the other key entities in a messaging environment are the external gateways required for each messaging channel. These gateways are not a part of UMS or Oracle WebLogic Server. Since UMS Drivers support widely-adopted messaging protocols, UMS can be integrated with existing infrastructures such as a corporate email servers or XMPP (Jabber) servers. Alternatively, UMS can connect to outside providers of SMS or text-to-speech services that support SMPP or VoiceXML, respectively.

11.1.2 Architecture

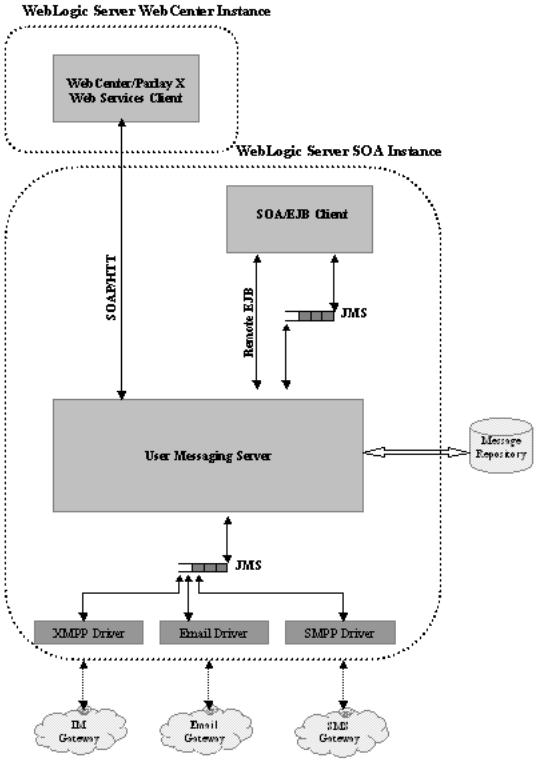
The system architecture of Oracle User Messaging Service is shown in Figure 11–1.

For maximum flexibility, the components of UMS are separate Java EE applications. This allows them to be deployed and managed independently of one another. For example, a particular driver can be stopped and reconfigured without affecting message delivery on all other channels.

Exchanges between UMS client applications and the UMS Server occur as SOAP/HTTP web service requests for web service clients, or through Remote EJB and JMS calls for BPEL messaging activities. Exchanges between the UMS Server and UMS Drivers occur through JMS queues.

Oracle UMS server and drivers are installed alongside SOA or BAM in their respective WebLogic Server instances. A WebCenter installation will include the necessary libraries to act as a UMS client application, invoking a server deployed in a SOA instance.

Figure 11–1 UMS architecture



11.2 Introduction to Oracle User Messaging Service Configuration

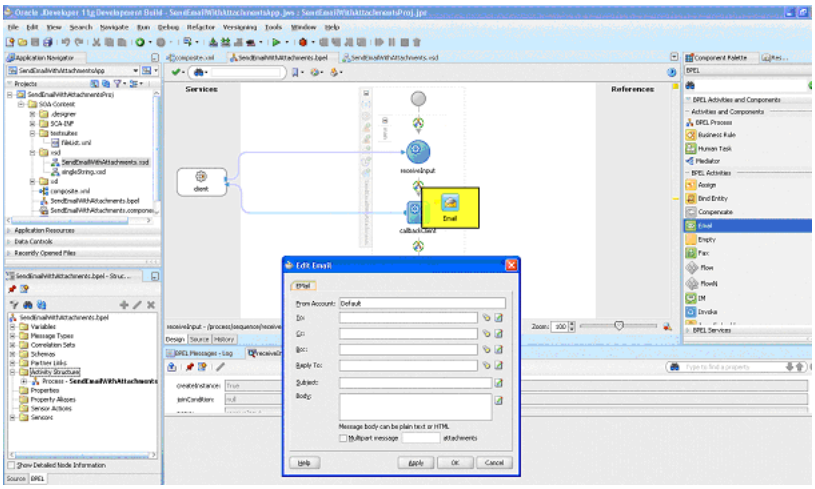
Oracle User Messaging Service enables users to receive notifications sent from SOA applications that are developed and deployed to the Oracle WebLogic Server using Oracle JDeveloper.

At the application level, there is notification activity for a specific delivery channel (such as SMS or E-Mail). For example, when you build a SOA application that sends e-mail notification, you drag and drop an *Email Activity* component from the JDeveloper *Component Palette* to the appropriate location within a workflow. The application connects then sends notifications.

For more information about Oracle JDeveloper, see your JDeveloper documentation.

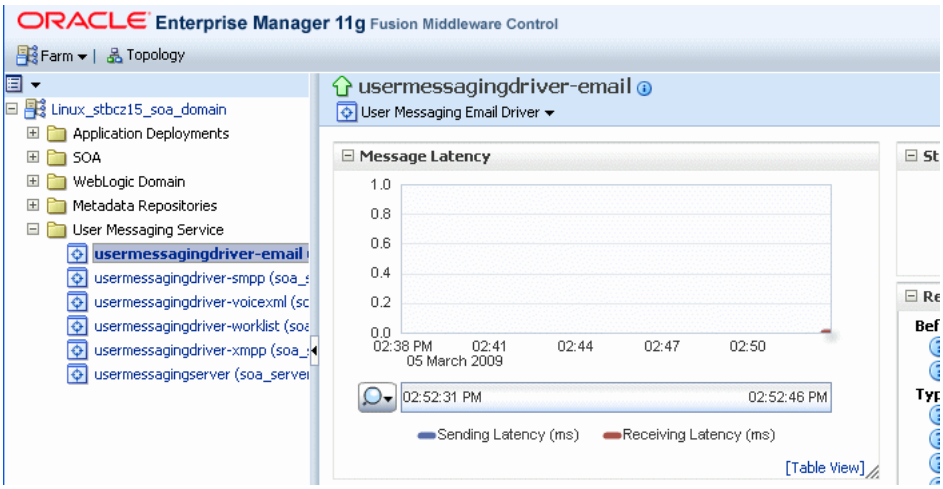
Figure 11–2 shows a user adding an Email Activity to the BPEL process of a SOA composite application.

Figure 11–2 Setting the Notification Activity in the BPEL Workflow



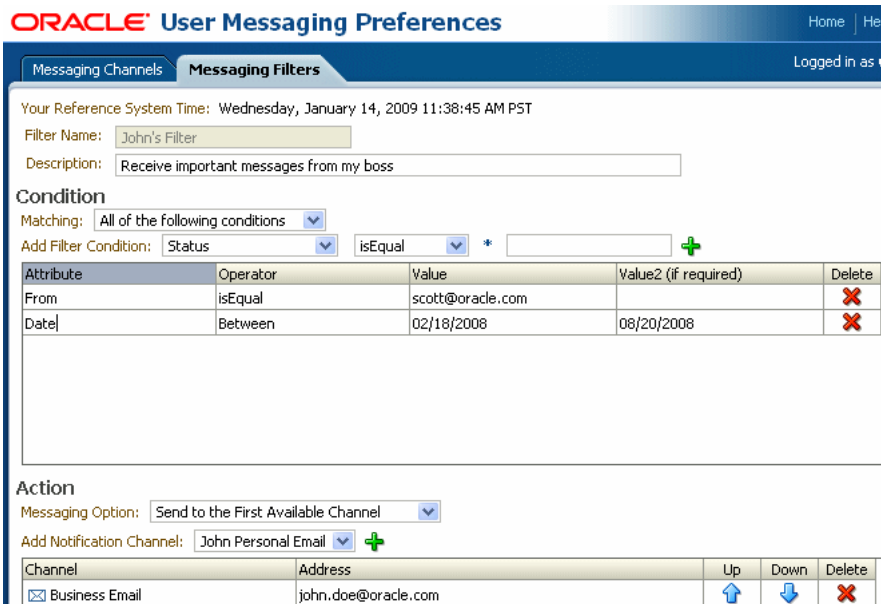
To enable the workflow participants to receive and forward notifications, use Oracle 11g Enterprise Manager to set the Oracle User Messaging Service environment by configuring the appropriate driver instances that reside on the same Oracle WebLogic Server on which you deploy the workflow application (Figure 11–3). Oracle User Messaging Service includes drivers that support messaging through E-Mail, IM, SMS and voice channels. For more information, see Section 11.4, "Configuring User Messaging Service Drivers".

Figure 11–3 Oracle Enterprise Manager 11g Fusion Middleware Control



In order for workflow participants to actually receive the notifications, they must register the devices that they use to access messages through User Messaging Preferences (Figure 11–4).

Figure 11–4 User Messaging Preferences



11.3 Accessing User Messaging Service Configuration Pages

You configure User Messaging Service through Oracle Enterprise Manager Fusion Middleware Control. For more information on Oracle Enterprise Manager, see your Oracle Enterprise Manager documentation.

11.3.1 How to Set the Storage Method

Use the Basic Configuration page to set deployment type for the Messaging Server (that is, select the storage method for run time and management data) and add (or remove) the User Messaging Preference Business Terms that are used for creating message filters.

Select Persistent (the default) to enable entries and the Messaging Store to persist when the server has been restarted. In the Transient mode (which is recommended for lightweight deployments), the Messaging Server does not maintain any data stored in the Messaging Store after a restart.

11.3.2 How to Add or Remove User Messaging Preferences Business Terms

The Basic Configuration page enables you to add or remove the business terms used to construct the message filters in User Message Preferences. For more information about building messaging filters with business terms, refer to Adding Business Terms.

11.3.2.1 Adding Business Terms

Note: Business Terms are stored per server instance. If there are multiple instances (as in a cluster), then new business terms must be added to each instance individually.

To add a business term to User Messaging Preferences:

1. Click **Add**.

2. Enter a descriptive name for the business term.
3. Select a data type (string, number, or date).
4. Click **Apply**.

11.3.2.2 Removing Business Terms

To remove a business term from User Messaging Preferences:

1. Select the business term.
2. Click **Delete**.
3. Click **Apply** to confirm the new term.

11.4 Configuring User Messaging Service Drivers

Oracle User Messaging Service includes the following drivers.

- E-Mail Driver
- SMPP Driver
- XMPP Driver
- Worklist Driver
- Proxy Driver

Note: For the cluster env, when you use separate messaging drivers for separate managed server nodes, all the drivers must be configured separately.

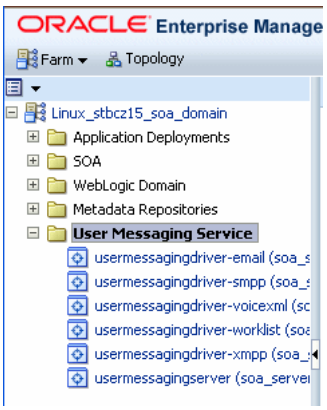
UMS Messaging Drivers are configured per instance. Configuring only one does not populate the configuration values to the drivers on the other cluster nodes.

11.4.1 How to Configure a Driver

To configure a driver:

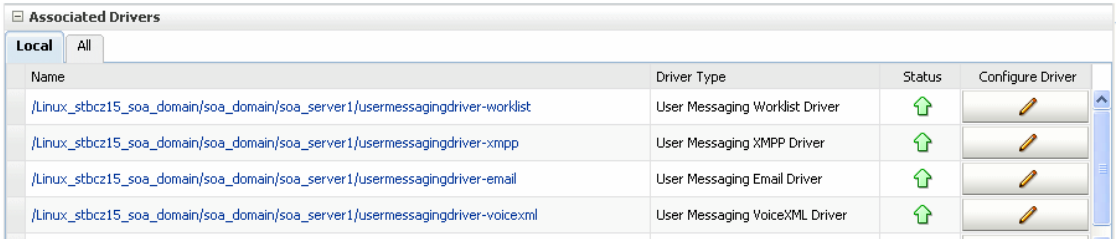
1. Log into the Enterprise Manager Fusion Middleware Control console as an administrator.
2. Expand the *Fusion Middleware* folder (Figure 11–5).

Figure 11–5 Expanding the UMS Folder



3. Navigate to the User Messaging Service *Home* page.
4. Click usermessagingserver(soa_server1). The Associated Drivers page appears.

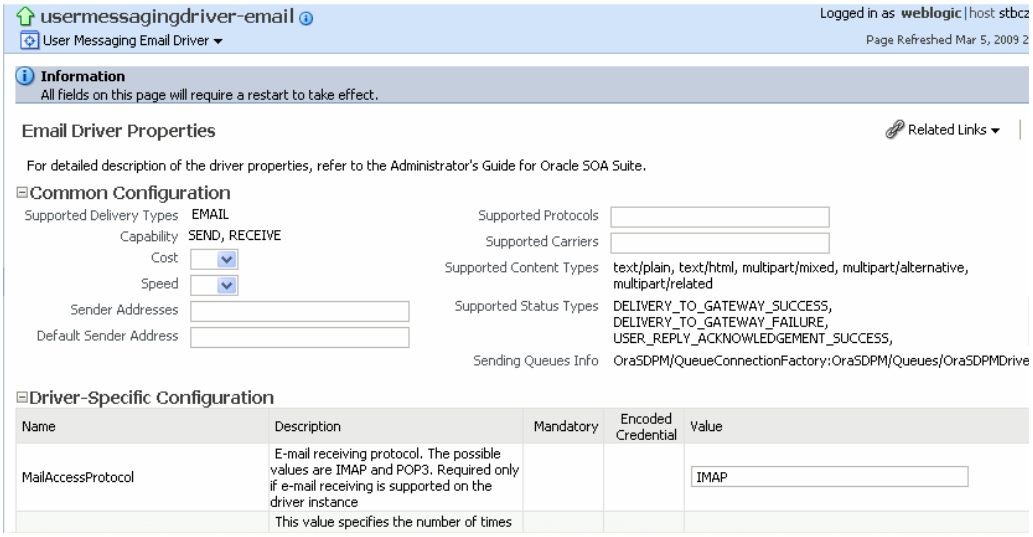
Figure 11–6 Drivers Associated with the UMS Instance



5. Select the *Local* tab to access the drivers collocated with the UMS server instance. These drivers may or may not be registered with the UMS server depending on whether or not they are properly configured. The *ALL* tab lists all drivers that are deployed in the domain and registered to all the UMS server instances.
6. Find the Email driver in the list, and then click the adjacent **Configure Driver** icon.

   The configuration page displays (Figure 11–7).

Figure 11–7 The Basic Configuration Page for a Selected Driver



7. If needed, expand the *Driver-Specific Configuration* section and configure the driver parameters. For more information, see Section 11.4.1.1, "About Driver Properties".

11.4.1.1 About Driver Properties

Oracle User Messaging Service drivers share common properties (listed in Table 11–1) that are used by the Messaging Engine when routing outbound messages. Typically, administrators set such Quality of Service (QoS) properties as driver cost (*Cost*) and driver speed (*Speed*), supported carriers (*SupportedCarriers*), and supported protocols (*SupportedProtocols*). Driver developers configure properties that typically do not require modification by the administrator, such as supported delivery types (*SupportedDeliveryTypes*), and supported content types (*SupportedContentTypes*).

Note: Properties such as *SendingQueuesInfo* are for advanced use and only require modification for advanced deployment topologies.

Table 11–1 Common Driver Properties

Name	Description	Mandatory Property?
Capability	Sets the driver's capability to send or receive messages. The values are *SEND*, *RECEIVE*, and *BOTH*.	Yes
Cost	The cost level of the driver (from 0 - 10). 0 is least expensive; 10 is most expensive. If the value is not in this range, cost is considered to be 0.	No
DefaultSenderAddress	The default address of the sender. The driver uses these addresses when sending a message that has no sender address specified, or when the specified sender address is not in the sender addresses list and the driver does not support using the application-provided sender address.	No
SenderAddresses	The list of sender addresses that the driver supports. If provided by the driver, the Messaging Engine can use this to route a sending message to the driver by matching against the sender address of the message.	No
SendingQueuesInfo	The information for the Driver Sending Queue.	Yes
Speed	The speed level of the driver (from 0-10, with 10 being the fastest).	No
SupportedCarriers	A comma-separated list of supported carriers.	No
SupportedContent Types	The content type supported by the driver.	Yes
SupportedDelivery Types	The delivery types supported by the driver.	Yes
SupportedProtocols	A comma-separated list of supported protocols. Entering an asterisk (*) for any protocol.	No
SupportedStatusTypes	The status types supported by the driver.	No
SupportsCancel	Supports a Cancel operation on a message.	No
SupportsReplace	Supports a Replace operation on a message.	No
SupportsStatusPolling	For certain protocols, an active polling of the remote gateway must be performed to check the status of a message previously sent. This property indicates whether the driver supports such status polling. If set to *true*, the Messaging Engine invokes the driver connection's `getStatus()` operation.	No
SupportsTracking	Supports Tracking operation on a message.	No

11.4.1.2 Securing Passwords

Sensitive driver properties (namely, passwords) can be stored securely in the credential store using Oracle Enterprise Manager. Properties are marked with the flag *Encoded Credential* and have a custom entry form field.

To store a sensitive driver property securely:

1. Go to the driver configuration page of the selected driver.
2. In the **Driver-Specific Configuration** section, locate the property with the *Encoded Credential* flag set.
3. Select the credential type (Depending on the selected credential type, you will be prompted to enter the username and/or password.). There are three options:

- Indirect password, create new user (*default option*)—specify the username and real password; the password will be stored in the credential store with the username as part of the key. The key and a fixed folder (*map name*) will be stored in the driver deployment's `driverconfig.xml`.
- Indirect password, use existing user—choose an existing username/key in the credential store (to reference the password you stored previously).
- User a clear text password—specify the password, and it will be stored directly in `driverconfig.xml`.

4. Click on **Apply** to save the changes.
5. Restart the driver application or the container for the changes to take effect.

You can check the password in the driver deployment directory's driverconfig.xml. For an indirect password, the format will be:

```
value="->mapName:keyName"    (mapName is the driver target name, and the key is
<parameter_name>.<username>)
```

For example, here is a sample entry in driverconfig.xml for an Email Driver's OutgoingPassword property:

```
<Property value="-&gt;/Farm_base_domain/base_domain/server_
soa/usermessagingdriver-email:OutgoingPassword.ouser" encodedCredential="true"
type="java.lang.String" mandatory="no" name="OutgoingPassword"
description="oracle.sdp.messaging.EmailDriverConfig.outgoingPassword"/>
```

11.4.1.3 Configuring the E-Mail Driver

The E-Mail Driver both sends and receives messages (that is, its *Capability* property is set to *BOTH* by default). The E-Mail Driver sends messages over SMTP and uses either IMAP and POP3 for receiving messages.

11.4.1.3.1 E-Mail Driver Interoperability This section details interoperability features of the E-Mail Driver.

The E-Mail driver is compatible with these protocols: POP3, IMAP4, and SMTP.

E-Mail Driver features include:

- Automatic connection retry
- SMTP for message sending
- IMAP4 and POP3 for message receiving (using polling)
- Scalable, highly available
- Prevents message loss and avoids duplication

The Gateway Vendors and Versions in Table 11–2 have been verified.

Table 11–2 E-Mail Driver Gateway Vendors and Versions

Vendor	Version
Oracle Beehive	Release 1 (1.4.3)
Oracle Collaboration Suite	10g Release 1 (10.1.2)
Microsoft Exchange	2003
Dovecot (IMAP4/POP3)	0.99.11
sendmail (SMTP)	8.13.1

11.4.1.3.2 Common Properties These are common driver properties that are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as SendingQueuesInfo are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values refer to the javadoc of DriverConfigPropertyNames.

Table 11–3 Common Email Properties

Name	Description	Mandatory	Default Value
InstanceName	Instance Name (for internal use only)	Yes	Email-Driver
Capability	Message sending and receiving capability	Yes	Both
SupportedDeliveryTypes	Supported Delivery Types	Yes	Email
SupportedContentTypes	Supported Content Types	Yes	text/plain, text/html, multipart/mixed, multipart/alternative, multipart/related
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE, USER_REPLY_ACKNOWLEDGEMENT_SUCCESS, USER_REPLY_ACKNOWLEDGEMENT_FAILURE
Cost	Cost	No	N/A
Speed	Speed	No	N/A
SupportedCarriers	Supported Carriers	No	N/A
Supported Protocols	Supported Protocols	No	N/A
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.3.3 Email Custom Properties These are properties specific to this driver and are generally associated with configuring access to the remote gateway and certain protocol or channel-specific behavior.

Table 11–4 Custom E-Mail Properties

Name	Description	Mandatory?	Default Value
MailAccessProtocol	E-mail receiving protocol. The possible values are IMAP and POP3. Required only if e-mail receiving is supported on the driver instance	No	IMAP
RetryLimit	This value specifies the number of times to retry connecting to the incoming mail server, if the connection is lost due to some reason. The default value is -1 which means no limit to the number of tries.	No	N/A
MailDelFreq	The frequency to permanently remove deleted messages. The unit is in seconds and the default value is 300 seconds. A negative value indicates the messages should not be expunged. For the POP3 protocol, the message is expunged after it is processed.	No	600
AutoDelete	This value indicates if the driver should mark the messages deleted after they have been processed. The value can be true or false and the default value is false. For the POP3 protocol, the messages are always deleted right after they are processed.	No	True
CheckMailFreq	The frequency with which to retrieve messages from the mail server. The unit is in seconds and the default value is 5 seconds.	No	30
ReceiveFolder	The name of the folder the driver is polling messages from. The default value is INBOX.	No	INBOX
OutgoingMailServer	The name of the SMTP server. Mandatory only if e-mail sending is required	No	N/A
OutgoingMailServerPort	The port number of SMTP server. Typically 25	No	25
OutgoingMailServerTLS	Whether to use TLS encryption to communicating to SMTP server.	No	False
OutgoingDefaultFromAddr	The default FROM address (if one is not provided in the outgoing message).	No	N/A
OutgoingUsername	The username used for SMTP authentication. Required only if SMTP authentication is supported by the SMTP server.	No	N/A
OutgoingPassword	The password used for SMTP authentication. Required only if SMTP authentication is supported by the SMTP server.	No	N/A
IncomingMailServer	The host name of the incoming mail server. Required only if e-mail receiving is supported on the driver instance.	No	N/A
IncomingMailServerPort	Port number of IMAP4 (i.e. 143 or 993) or POP3 (i.e. 110 or 995) server.	No	N/A
IncomingMailServerSSL	Whether to enable SSL when connecting to IMAP4 or POP3 server.	No	False

Table 11–4 (Cont.) Custom E-Mail Properties

Name	Description	Mandatory?	Default Value
IncomingMailIDs	The e-mail addresses corresponding to the user names. Each e-mail address is separated by a comma and must reside in the same position in the list as their corresponding user name appears on the usernames list. Required only if e-mail receiving is supported on the driver instance.	No	N/A
IncomingUserIDs	The list of user names of the mail accounts the driver instance is polling from. Each name must be separated by a comma, for example, foo,bar. Required only if e-mail receiving is supported on the driver instance	No	N/A
IncomingUserPasswords	The list of passwords corresponding to the user names. Each password is separated by a comma and must reside in the same position in the list as their corresponding user name appears on the usernames list. Required only if e-mail receiving is supported on the driver instance.	No	N/A
IncomingProcessingChunkSize	Max number of messages processed per message polling.	No	100

11.4.1.3.4 Client API MessageInfo Support These properties are message delivery related which are specified through client API. Table 11–5 describes if the protocol or driver implementation honors such properties.

Table 11–5 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	False
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.4.1.4 Configuring the SMPP Driver

SMPP (Short Message Peer-to-Peer) is one of the most popular GSM SMS protocols. User Messaging Service includes a pre-built implementation of the SMPP protocol as a driver that is capable of both sending and receiving short messages. If the sending feature is enabled, the SMPP driver opens one TCP connection to the SMS-C (Short Message Service Center) as a transmitter for sending. If the driver's receiving feature is enabled, it opens another connection to the SMS-C as a receiver for receiving. Only two TCP connections (both initiated by the driver) are needed for all communication between the driver and the SMS-C.

Note: The SMPP Driver implements Version 3.4 of the SMPP protocol and only supports connections to an SMS-C that supports this version.

11.4.1.4.1 SMPP Driver Interoperability This section details interoperability features of the SMPP Driver.

The SMPP driver is compatible with these protocols: SMPP v3.4.

SMPP Driver features include:

- Automatic connection retry
- HTTP proxy for firewall traversal
- Authentication configuration
- Configurable chunk size
- Bulk Sending
- Encoding: UCS2, IA5, GSM_DEFAULT
- Priority Setting
- Configurable Window size
- Plain text content only

The Gateway Vendors in Table 11–6 have been verified.

Table 11–6 SMPP Driver Gateway Vendors

Vendor
Logica CMG
Clickatell
Verisign
OpenSMPP (simulator)

11.4.1.4.2 Common Properties These are common driver properties that are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as SendingQueuesInfo are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values refer to the javadoc of DriverConfigPropertyNames.

Table 11–7 Common SMPP Properties

Name	Description	Mandatory	Default Value
InstanceName	Instance Name (for internal use only)	Yes	SMPP-Driver
Capability	Message sending and receiving capability	Yes	Both
SupportedDeliveryTypes	Supported Delivery Types	Yes	SMS
SupportedContentTypes	Supported Content Types	Yes	text/plain
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_ GATEWAY_SUCCESS, DELIVERY_TO_ GATEWAY_FAILURE
Cost	Cost	No	N/A
Speed	Speed	No	N/A
SupportedCarriers	Supported Carriers	No	N/A
Supported Protocols	Supported Protocols	No	N/A

Table 11–7 (Cont.) Common SMPP Properties

Name	Description	Mandatory	Default Value
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.4.3 Custom Properties These are properties specific to this driver and are generally associated with configuring access to the remote gateway and certain protocol or channel-specific behavior.

Table 11–8 Custom SMPP Properties

Name	Description	Mandatory?	Default Value
SmsAccountId	The Account Identifier on the SMS-C.	Yes	N/A
SmsServerHost	The name (or IP address) of the SMS-C server.	Yes	N/A
TransmitterSystemId	The account ID that is used to send out messages.	Yes	N/A
ReceiverSystemId	The account ID that is used to receive messages.	Yes	N/A
TransmitterSystemType	The type of transmitter system.	Yes	The default value is *Logica*.
ReceiverSystemType	The type of receiver system.	Yes	The default value is *Logica*.
TransmitterSystemPassword	The password of the transmitter system.	Yes	N/A
ReceiverSystemPassword	The password for the receiver system.	Yes	N/A
ServerTransmitterPort	The TCP port number of the transmitter system.	Yes	N/A
ServerReceiverPort	The TCP port number of the receiver system.	Yes	N/A
DefaultEncoding	The default encoding of the SMPP driver.	No	The default value is *UCS2*.
EncodingAutoDetect	If set to *true* (the default), the SMPP driver encodes automatically.	No	The default value is *true*.
LocalSendingPort	The local TCP port used by the SMPP driver to messages to the SMS-C.	No	N/A
LocalReceivingPort	The local TCP port used by the SMPP drivers to receive messages from the SMS-C.	No	N/A

Table 11–8 (Cont.) Custom SMPP Properties

Name	Description	Mandatory?	Default Value
LocalAddress	The host name (or IP address) of the server that hosts the SMPP driver.	No	N/A
WindowSize	The window size for SMS. This value must be a positive number.	No	The default value is 1.
EnquireInterval	The interval, in seconds, to send an enquire message to the SMS-C.	No	The default value is 30.
ThrottleDelay	The delay, in seconds, between throttles.	No	The default value is 15.
BindRetryDelay	The delay, in seconds, for a binding retry.	No	The default value is 30.
ResponseTimer	Time lapse allowed between SMPP request and response, in seconds. Default is 30.	No	30
RegisteredDeliveryMask	The delay, in seconds, for a binding retry.	No	0xFF
RangeSetNull	Set to *true* to set the *address range* field of BIND_RECEIVER to *null*. Set to *false* (the default value) to set the address range field to *SmsSystemId*.	No	The default value is *false*.
PriorityAllowed	The highest priority allowed for the SMPP driver. The range is 0 (normal) to 3 (highest).	No	The default value is 0.
BulkSending	Setting this value to *true* (the default) enables sending messages in bulk to the SMS-C.	No.	The default value is *true*.
PayloadSending	If set to true, the SMPP driver always uses the message payload properties when sending messages to the SMS-C.	No	The default value is false.
SourceTon	The Type of Number (TON) for ESME address(es) served through SMPP receiver session.	No	The default value is 0.
SourceNpi	The Numbering Plan Indicator (NPI) for ESME address(es) served through the SMPP receiver session.	No	The default value is 0.
DestinationTon	The Type of Number (TON) for destination.	No	The default value is 0.
DestinationNpi	The Numbering Plan Indicator (NPI) for destination.	No	The default value is 0.
ExtraErrorCode	A comma-separated list of error codes.	No	N/A
MaxChunks	The maximum SMS chunks for a message.	No	The default value is -1 (no maximum).
ChunkSize	The size of each SMS message chunk.	No	The default value is 160.
LongMessageSending	Supports sending long messages.	No	N/A
DatagramMessageMode	Supports Datagram Message mode.	No	N/A

11.4.1.4.4 Client API MessageInfo Support These properties are message delivery related which are specified through client API. Table 11–9 describes if the protocol or driver implementation honors such properties.

Table 11–9 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	True
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.4.1.5 Configuring the XMPP Driver

The XMPP Driver provides unidirectional as well as bidirectional access from Oracle Fusion Middleware to end users for real-time instant messaging (IM) via XMPP (Extensible Messaging and Presence Protocol). This driver enables end users to receive alert notifications or interactively chat with applications through their IM client of choice.

11.4.1.5.1 About XMPP XMPP is an open, XML-based protocol for Instant Messaging and Presence. XMPP-based software is deployed on thousands of servers across the Internet and is used by millions of people worldwide. XMPP consists of a client-server architecture, which resembles the ubiquitous e-mail network. XMPP servers are completely decentralized, allowing anyone to set up their own server. Messaging is achieved as in the email network, where recipients are addressed by a username and a host name (for example: username@host name). In the XMPP network, users are identified by an XMPP (Jabber) ID, which consists of a username and the host name of the particular XMPP server to which the user connects. An end user of XMPP connects to an XMPP server using an XMPP client in order to send instant messages to other XMPP users. XMPP, however, is not the only protocol network available for instant messaging. XMPP has an extensible and modular architecture. It integrates with proprietary IM networks such as Yahoo, MSN, AOL and ICQ using transport gateways that can connect to these networks. This allows XMPP users to communicate with those on other networks.

In order to use the XMPP Driver in UMS, you must have access to a Jabber/XMPP server and an XMPP account for the UMS XMPP Driver instance to login as. In addition, the XMPP Driver includes configuration parameters that enable UMS to communicate with users on Yahoo, MSN, AOL or ICQ IM networks. This requires that you additionally have accounts on these proprietary IM networks to which you are connecting from the XMPP Driver, and thus, allow end users of those particular networks to communicate with UMS.

11.4.1.5.2 XMPP Driver Interoperability This section details interoperability features of the XMPP Driver.

The XMPP driver is compatible with these protocols: XMPP (RFC 3920, 3921).

XMPP Driver features include:

- Automatic connection retry
- HTTP proxy for firewall traversal
- Plain text content only

The Gateway Vendors and Versions in Table 11–6 have been verified.

Table 11–10 XMPP Driver Gateway Vendors and Versions

Vendor	Version
Jabberd	v1, v2

Table 11–10 (Cont.) XMPP Driver Gateway Vendors and Versions

Vendor	Version
ejabberd	v2

11.4.1.5.3 Third-Party Software The XMPP Driver uses or requires the following third-party software:

Table 11–11 Required Third-Party Software

Name	Instructions	Version(s)
JabberBeans	This driver uses the JabberBeans Java library to connect to a Jabber/XMPP Instant Messaging Server. This driver includes a licensed copy of JabberBeans (version 0.9.1).	0.9.1
XMPP Server	Optional. To download and install your own Jabber/XMPP server, pick and install a server from http://www.jabber.org.	
Yahoo, MSN, AOL(AIM), and ICQ Transport Gateways	Optional. Follow the transport installation guide that comes with the Jabber/XMPP server to install and configure one or more transports to connect to proprietary IM gateways.	

Note: You do not need to install your own XMPP Server if you have access to an existing server. For a list of public servers, see http://www.jabber.org.

11.4.1.5.4 Driver Application Archive (EAR) `$ORACLE_HOME/communications/applications/sdpmessagingdriver-xmpp.ear`

11.4.1.5.5 Common Properties These are common driver properties that are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as SendingQueuesInfo are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values refer to the javadoc of DriverConfigPropertyNames.

Table 11–12 Common XMPP Properties

Name	Description	Mandatory	Default Value
InstanceName	Instance Name (for internal use only)	Yes	XMPP-IM-Driver
Capability	Message sending and receiving capability	Yes	Both
SupportedDeliveryTypes	Supported Delivery Types	Yes	IM
SupportedContentTypes	Supported Content Types	Yes	text/plain

Table 11–12 (Cont.) Common XMPP Properties

Name	Description	Mandatory	Default Value
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_ GATEWAY_SUCCESS, DELIVERY_TO_ GATEWAY_FAILURE
Cost	Cost	No	N/A
Speed	Speed	No	N/A
SupportedCarriers	Supported Carriers	No	N/A
Supported Protocols	Supported Protocols	No	N/A
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.5.6 XMPP Custom Properties The XMPP Driver includes the custom properties shown below.

Table 11–13 Custom XMPP Properties

Name	Description	Mandatory	Default Values
IMServerHost	Jabber server host name. For multiple servers, use a comma-separated list (for example: `my1.host.com, my2.host.com`). If only one host name is specified, it will be used for all accounts.	Yes	N/A
IMServerPort	Corresponding comma-separated list of Jabber server ports (e.g. 5222, 5222)	Yes	5222
IMServerUsername	List of Jabber usernames to login as (these user accounts will be automatically created, if necessary, on the corresponding Jabber servers). If you have multiple servers listed above, there must be an equal number of usernames (one username per server). If you have only one server listed above, all usernames listed here will use that server (e.g. oracleagent1, oracleagent2). You may also enter a complete Jabber ID if its domain name is different from the Jabber server host name (for example: `oracleagent1@host.com`).	Yes	N/A
IMServerPassword	Corresponding comma-separated list of passwords for each username listed above.	Yes	N/A

Table 11–13 (Cont.) Custom XMPP Properties

Name	Description	Mandatory	Default Values
YahooEnable	Enable/disable Yahoo Transport (set `true` to *enable*, and leave blank to set `false` to *disable*), for each user account specified above in a comma-separated list.	No	N/A
YahooUsername	Comma-separated list of Yahoo account IDs (requires that you already have these IDs registered on Yahoo), for each user account above (leave entries blank for accounts without Yahoo). Entering valid Yahoo account info allows Yahoo users to access applications via instant messaging.	No	N/A
YahooPassword	Corresponding comma-separated list of Yahoo account passwords.	No	N/A
MSNEnable	Enable/Disable MSN Transport (set 'true' to enable, and leave blank or set 'false' to disable), for each user account specified above in a comma-separated list.	No	N/A
MSNUsername	Comma-separated list of MSN Messenger (known as .NET passport) account IDs (requires that you already have these IDs registered as .NET passports), for each user account above (leave entries blank for accounts without MSN). Entering valid .NET account info allows MSN Messenger users to access applications via instant messaging.	No	N/A
MSNPassword	Corresponding comma-separated list of MSN Messenger account passwords.	No	N/A
AOLEnable	Enable/Disable AOL IM (AIM) Transport (set 'true' to enable, and leave blank or set 'false' to disable), for each user account specified above in a comma-separated list.	No	N/A
AOLUsername	Comma-separated list of AOL IM (AIM) account IDs (requires that you already have these IDs registered with AOL), for each user account above (leave entries blank for accounts without AOL). Entering valid AOL account info allows AOL users to access applications via instant messaging.	No	N/A
AOLPassword	Corresponding comma-separated list of AOL IM account passwords.	No	N/A
ICQEnable	Enable/Disable ICQ IM Transport (set 'true' to enable, and leave blank or set 'false' to disable), for each user account specified above in a comma-separated list.	No	N/A
ICQUsername	Comma-separated list of ICQ account IDs (requires that you already have these IDs registered with ICQ), for each user account above (leave entries blank for accounts without ICQ). Entering valid ICQ account info allows ICQ users to access applications via instant messaging	No	N/A
ICQPassword	Corresponding comma-separated list of ICQ account passwords.	No	N/A
RetryLimit	Number of times the driver should attempt to reconnect when disconnected from the Jabber server. Enter -1 for unlimited retries.	No	N/A

Table 11–13 (Cont.) Custom XMPP Properties

Name	Description	Mandatory	Default Values
RetryInterval	Time interval (in seconds) between reconnect attempts.	No	N/A

11.4.1.5.7 Client API MessageInfo Support These properties are message delivery related which are specified through client API. The table below describes if the protocol or driver implementation honors such properties.

Table 11–14 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	False
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.4.1.6 Configuring the VoiceXML Driver

The VoiceXML Driver supports the Genesys VoiceGenie gateway's outbound call protocol to send messages authored in VoiceXML. The gateway delivers the message using text-to-speech synthesis.

11.4.1.6.1 VoiceXML Driver Interoperability This section details interoperability features of the VoiceXML Driver.

The VoiceXML driver is compatible with these protocols: VoiceXML over HTTP (VoiceGenie gateway protocol).

VoiceXML Driver features include:

- VoiceXML content only

The Gateway Vendor and Version in Table 11–6 has been verified.

Table 11–15 VoiceXML Driver Gateway Vendor and Version

Vendor	Version
Genesys VoiceGenie	6.4.2

11.4.1.6.2 Common Properties These are common driver properties that are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as SendingQueuesInfo are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values refer to the javadoc of DriverConfigPropertyNames.

Table 11–16 Common VoiceXML Properties

Name	Description	Mandatory	Default Value
InstanceName	Instance Name (for internal use only)	Yes	VoiceXML-Driver

Table 11–16 (Cont.) Common VoiceXML Properties

Name	Description	Mandatory	Default Value
Capability	Message sending and receiving capability	Yes	SEND
SupportedDeliveryTypes	Supported Delivery Types	Yes	VOICE
SupportedContentTypes	Supported Content Types	Yes	text/vxml, text/x-vxml
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	Cost	No	N/A
Speed	Speed	No	N/A
SupportedCarriers	Supported Carriers	No	N/A
Supported Protocols	Supported Protocols	No	N/A
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.6.3 VoiceXML Custom Properties The VoiceXML Driver includes the custom properties shown below.

Table 11–17 Custom VoiceXML Properties

Name	Description	Mandatory	Default Values
VoiceXMLOutboundServletURI	The URL of the VoiceXML/VoiceGenie gateway.	Yes	N/A
VoiceXMLOutboundServletUserName	The user name of the VoiceXML gateway.	No	N/A
VoiceXMLOutboundServletPassword	The password of the VoiceXML gateway.	No	N/A
VoiceXMLOutboundServletDNIS	The number that appears in the recipient's ID display.	No	N/A

Table 11–17 (Cont.) Custom VoiceXML Properties

Name	Description	Mandatory	Default Values
VoiceXMLReceiveURL	The URL of this driver's servlet which will handle incoming requests from the VoiceXML Gateway. The format is `http://<host>:<port>/usermessagingdriver-voicexml/receive`. The default behavior, if this property is not set, is to use the local container's HTTP listen host and port. The auto-generated default URL will only work for the first driver instance. For additional instances, the context root is different and this property must be configured using the correct context root replacement for `/sdpmessagingdriver-voicexml`.	No	N/A

11.4.1.6.4 Client API MessageInfo Support These properties are message delivery related which are specified through client API. The table below describes if the protocol or driver implementation honors such properties.

Table 11–18 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	False
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.4.1.7 Configuring the Worklist Driver

The Worklist driver enables notifications from all sources to be sent to users in the form of *worklist* tasks for integration into the users' WebCenter Unified Worklist.

> **Note:** Worklist Message tasks are accessible both through a WebCenter that has been configured to search the BPEL connection that the Worklist message driver is sending messages to, as well as through the BPEL Worklist application. The BPEL Worklist Application will also show these message-based tasks as Worklist items.

This integration is achieved by exposing a *Worklist* channel (delivery type) to applications and end users. Messages sent through the user's Worklist channel are processed by the Worklist driver. The User Messaging Service API semantics are the same as those for existing channels such as IM or Email. *This driver handles sending messages only.* The Driver Application Archive (EAR) is located at: `$ORACLE_HOME/communications/applications/sdpmessagingdriver-worklist.ear`

11.4.1.7.1 Install the Worklist Driver To enable the messaging worklist feature, the WebLogic SOA domain must be extended using the extension template available at `$ORACLE_HOME/common/templates/applications/oracle.ums.driver.worklist_template_11.1.1.jar`. To extend a SOA domain using the Oracle Fusion Middleware Configuration Wizard:

1. Launch Oracle Fusion Middleware Configuration Wizard (`$ORACLE_HOME/common/bin/config.sh or %ORACLE_HOME%\common\bin\config.cmd`).
2. Select the *Extend an existing WebLogic domain* option.
3. Select the desired SOA domain directory.
4. Select the *Extend my domain using an existing extension template* option.
5. Click **Browse**, and navigate to `$ORACLE_HOME/common/templates/applications`
6. Select oracle.ums.driver.worklist_template_11.1.1.jar
7. Complete the remaining steps of the Oracle Fusion Middleware Configuration Wizard, and restart the SOA servers.

Note: *Special Considerations if the SOA managed server is on a remote machine*: The `oracle.ums.driver.worklist_template_11.1.1.jar` extension template includes a SOA composite application (`sca_sdpmessagingsca-worklist-composite_rev1.0.jar`) that is copied to `$DOMAIN_HOME/soa/autodeploy`, and is auto-deployed by the SOA Infra runtime upon server restart. However, if the SOA Infra runtime is on a remote machine, and the domain is packed with the *-managed=true* option (the correct option to use), this directory is not included in the archive. Thus, the composite is not deployed upon restarting the SOA managed server.

In order to complete the installation, copy the contents of `$DOMAIN_HOME/soa/autodeploy` from the AdminServer machine to the corresponding location on the remote machine with the SOA managed server, and restart the SOA managed server. You may have to create the directory structure `soa/autodeploy` under `$DOMAIN_HOME` on the remote machine.

11.4.1.7.2 Common Properties The following common driver properties are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as `SendingQueuesInfo` are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values see the javadoc of `DriverConfigPropertyNames`.

Table 11–19 Common Worklist Properties

Name	Description	Mandatory?	Default Value
InstanceName	Instance Name (for internal use only)	Yes	Worklist-Driver
Capability	Message sending and receiving capability	Yes	SEND
SupportedDeliveryTypes	Supported Delivery Types	Yes	WORKLIST
SupportedContentTypes	Supported Content Types	Yes	text/plain, text/html
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	Cost	No	N/A

Table 11–19 (Cont.) Common Worklist Properties

Name	Description	Mandatory?	Default Value
Speed	Speed	No	N/A
SupportedCarriers	SupportedCarriers	No	N/A
SupportedProtocols	SupportedProtocols	No	N/A
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.7.3 Custom Properties The following custom property is available:

Table 11–20 Custom Worklist Property

Name	Description	Mandatory	Default Value
BPELConnectionURL	The URL of the BPEL server to connect to. The format is `http://<bpel-host>:<bpel-port>`. The default behavior, unless changed, is to use the local container's HTTP connection URL.		

11.4.1.7.4 Client API MessageInfo Support This table shows if the protocol or driver implementation honor the following message delivery-related properties that are specified through the client API.

Table 11–21 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	False
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.4.1.8 Configuring the Proxy Driver

The Proxy Driver acts as a Messaging Web Service client to a Fusion Middleware Messaging server hosted elsewhere in the intranet or Internet. It uses SOAP over HTTP (the Parlay X Multimedia Web Service protocol) to send messages and receive messages as well as return message delivery status. The ParlayX Web Service relays messages from one UMS instance to another. It can be used to relay traffic from multiple instances in an Intranet to a terminating instance that has all of the protocol-specific drivers configured to an external gateway such as an SMSC, or to an SMTP or IMAP mail server.

11.4.1.8.1 Common Properties These are common driver properties that are indicative of the capabilities of this driver for use by the engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Some properties such as SendingQueuesInfo are for advanced use and only require modification for advanced deployment topologies. For a complete description of these properties and available values refer to the javadoc of DriverConfigPropertyNames.

Table 11–22 Common Proxy Properties

Name	Description	Mandatory	Default Value
InstanceName	Instance Name (for internal use only)	Yes	Proxy-Driver
Capability	Message sending and receiving capability	Yes	SEND
SupportedDeliveryTypes	Supported Delivery Types	Yes	EMAIL, SMS, VOICE, IM, WORKLIST
SupportedContentTypes	Supported Content Types	Yes	*
SupportedStatusTypes	Supported Status Types	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	Cost	No	N/A
Speed	Speed	No	N/A
SupportedCarriers	Supported Carriers	No	N/A
Supported Protocols	Supported Protocols	No	N/A
SupportsCancel	Supports Cancel Operation on the Message	No	False
SupportsReplace	Supports Replace Operation on the Message	No	False
SupportsTracking	Supports Tracking Operation on the Message	No	False
SupportsStatusPolling	Supports Status Polling Operation on the Message	No	False
SenderAddresses	Sender Addresses	No	N/A
DefaultSenderAddress	Default Sender Address	No	N/A
SendingQueuesInfo	Driver Sending Queue Info	Yes	OraSDPM/QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1

11.4.1.8.2 Proxy Custom Properties The Proxy Driver includes the custom properties shown below.

Table 11–23 Custom Proxy Properties

Name	Description	Mandatory	Default Values
GatewayURL	The URL to the hosted 11*g* UMS Web Service gateway. The URL is in the following format: `http://<host>:<port>/sdpmessaging/parlayx/SendMessageService`	Yes	N/A
Username	Username of the messaging gateway.	No	N/A
Password	The password of the username	No	N/A
Policies	Comma-separated list of Oracle Web Services Manager WS-Security policies to be attached to proxy driver requests	No	N/A

11.4.1.8.3 Client API MessageInfo Support These properties are message delivery related which are specified through client API. The table below describes if the protocol or driver implementation honors such properties.

Table 11–24 Client API MessageInfo Support

Name	Description	Support
Expiration	*Expiration* means how long the message will exist until it expires.	False
Delay	*Delay* means the amount of time that must elapse before the message is sent.	False

11.5 Securing User Messaging Service

The User Messaging Preferences User Interface and the Parlay X Web Services can be secured at the transport-level using Secure Sockets Layer (SSL). By default, all deployed web services are unsecured. Web Service Security should be enabled for any services that will be deployed in a production environment.

- To enable SSL in the Oracle WebLogic Server, see "Configure SSL for Oracle WebLogic Server" in the *Oracle Fusion Middleware Administrator's Guide*. This step is sufficient to secure the User Messaging Preferences User Interface.
- To secure the Parlay X Web Services, see "Configuring Transport-Level Security" in the *Securing WebLogic Web Services*.

UMS supports the use of Oracle Web Services Manager WS-Security policies to protect UMS web services. For more information about Oracle Web Services Manager, see "Using Oracle Web Service Security Policies", in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*.

The recommended security configuration for web services uses Security Assertion Markup Language (SAML) tokens to pass identities between web service clients and UMS. With SAML tokens, instead of the web service client passing a username and password to UMS, a trust relationship is established between the client and UMS by means of exchanging certificates. Once this keystore configuration is in place, the web service client passes only the user identity, and vouches for the fact that it has authenticated the user appropriately.

The recommended policies to use for UMS web services are:

- `oracle/wss11_saml_token_with_message_protection_service_policy (server-side)`

- `oracle/wss11_saml_token_with_message_protection_service_policy (client-side)`

11.5.1 Web Service Security on Notification

The different Web services include corresponding notification Web services (MessageNotification, PresenceNotification) that run on the client side and receive notifications (message delivery status, message receipt, presence status change) when the appropriate event occurs. This implementation does not provide for the use of Web Service security (WS-Security) by default during notification of the clients. That is, the server assumes that the notification Web services running on the client side do not use WS-Security, and makes no attempt to authenticate itself when sending notifications. If you enable WS-Security on the client side, the notification from the server will fail because the notification SOAP request will be missing the required headers.

11.5.2 Enabling UMS Service Security

To enable a policy for an UMS web service, follow the steps in "Configuring Oracle WSM Security Policies in Administration Console" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*, selecting policy `oracle/wss11_saml_token_with_message_protection_service_policy`. This configuration must be repeated for each service that you wish to secure.

11.5.3 Enabling Client Security

Web service client security must be enabled programmatically. When using the client libraries described in *Parlay X Messaging Client API and Client Proxy Packages* (in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*), WS-Security policy configuration is provided when a client object is constructed. The client constructors take an argument of type `Map<String, Object>`. In general when using SAML authentication, the key/value pairs (Table 11–25) should be added to the configuration map in addition to other required properties such as the endpoint address.

Table 11–25 Client security keys

Key	Type	Typical Value
`oracle.sdp.parlayx.ParlayXConstants.POLICIES`	String[]	`oracle/wss11_saml_token_with_message_protection_client_policy`
`javax.xml.ws.BindingProvider.USERNAME_PROPERTY`	String	*`<valid username>`*
`oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_ALIAS_PROPERTY`	String	(optional) keystore alias for target service. See Client Aliases.

Example 11–1 Web Service Client Security

```
import oracle.sdp.parlayx.presence.consumer.PresenceConsumerClient;

...

Map<String, Object> config = new HashMap<String, Object>();
config.put(javax.xml.ws.BindingProvider.ENDPOINT_ADDRESS_PROPERTY, ums_url);
config.put(oracle.sdp.parlayx.ParlayXConstants.POLICIES, new String[]
{"oracle/wss11_saml_token_with_message_protection_client_policy"});
```

```
config.put(javax.xml.ws.BindingProvider.USERNAME_PROPERTY, "test.user1");

PresenceConsumerClient presenceClient = new PresenceConsumerClient(config);
```

11.5.4 Keystore Configuration

In order to use the recommended WS-Security policy, you must configure a keystore containing the public and private key information required by OWSM. Refer to "Configuring the Credential Store Using WLST" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server* for information on how to configure the keystore and corresponding credential store entries.

- If both your web service client and UMS server are in the same domain, then they share a keystore and credential store.
- If your web service client and UMS server are in different domains, then you must import the UMS public key into your client domain's keystore, and must import your client domain's public key into the UMS keystore.

11.5.5 Client Aliases

When using certain WS-Security policies such as the SAML policy recommended here, the client must use the server's public key to encrypt the web service request. However, there is generally only one keystore configured per domain. Therefore, if you have a domain in which there are web service clients that communicate with web services in multiple other domains, then you may need to override the default keystore entry used by OWSM.

For example, if you have a domain in which application "A" is a web service client to a UMS web service, and application "B" is a web service client to a web service in another domain, then A's requests must be encrypted using the public key of the UMS domain, and B's requests must be encrypted using the public key of the other domain. You can accomplish this goal by overriding the keystore alias used by OWSM for each request:

- Import (for example) the UMS public key with alias "ums_public_key", and the other public key with alias "other_public_key".
- When creating an UMS web service client, specify the recipient keystore alias parameter, setting the key to `oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_ALIAS_PROPERTY` and the value to "ums_public_key" as shown in Example 11–2.

Example 11–2 Client Aliases

```
import oracle.sdp.parlayx.multimedia_messaging.send.SendMessageClient

...

Map<String, Object> config = new HashMap<String, Object>();
config.put(javax.xml.ws.BindingProvider.ENDPOINT_ADDRESS_PROPERTY, ums_url);
config.put(oracle.sdp.parlayx.ParlayXConstants.POLICIES, new String[]
{"oracle/wss11_saml_token_with_message_protection_client_policy"});
config.put(javax.xml.ws.BindingProvider.USERNAME_PROPERTY, "test.user1");
config.put(oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_
ALIAS_PROPERTY, "ums_public_key")
SendMessageClient sendClient = new SendMessageClient(config);
```

- The other web service client will similarly need to override the keystore alias, but the exact mechanism may differ. For example if using a JAX-WS client stub directly, then you can add the override property to the JAX-WS request context. See "Policy Configuration Overrides for the Web Service Client" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server* for more details.

11.6 Troubleshooting Oracle User Messaging Service

To debug User Messaging Service, first check the server diagnostic logs. The logs may contain exception, error, or warning messages that provide details about incorrect behavior along with actions to remedy the problem. The following table describes additional methods for debugging common User Messaging Service problems.

Table 11–26 Troubleshooting UMS

Symptom	Possible Causes	Solutions
Notifications are not being sent from BPEL or Human Workflow components in SOA.	Notification Mode is set to NONE in SOA Workflow Notification configuration.	Change the Notification Mode setting to *EMAIL* or *ALL* using Oracle Fusion Middleware Control.
Email notification is not being sent.	The Outgoing (SMTP) Mail Server settings in the UMS Email Driver are incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `OutgoingMailServer` ■ `OutgoingMailServerPort` Note: Validate the values by using them in any e-mail client for connecting to the SMTP server.
	The SMTP server requires authentication or a secure connection (TLS or SSL).	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `OutgoingUsername` ■ `OutgoingPassword` ■ `OutgoingMailServerSecurity`

Table 11–26 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions

Notifications are not being sent because of error message: `No matching drivers found for sender address = <address>`	The UMS Driver for the appropriate channel is configured with a specific list of *SenderAddresses*, and the message sent by the application has set a non-matching Sender Address.
	
	Note: UMS Server matches the outbound message's sender address, if set, against the available drivers' *SenderAddresses* to find a matching driver to use for delivering the message. If a driver has set one or more SenderAddresses, then the UMS Server will only send messages with the matching sender address to it.
		■ Check the following settings in the appropriate UMS Driver using Oracle Fusion Middleware Control:
		`SenderAddresses`
		Note: The format for SenderAddresses is a comma-separated list of `<DeliveryType>:<Address>`.
		For example:
		`EMAIL:sender@example.com, EMAIL:sender@example2.com`
		■ Leave this property blank, if you want this driver to service outbound messages for all sender addresses for this channel (delivery type).
		■ If there are multiple driver instances deployed for the same channel (delivery type) with different configurations, use the *SenderAddresses* to differentiate the driver instances. For example, one instance can be set with a specific value in SenderAddresses to only service outbound messages with that matching sender address, while the other instance can keep the SenderAddresses blank in order to service all outbound messages that do not specify any sender address or one that does not match that of the first driver instance.
		■ SenderAddresses that are configured with the incorrect syntax (such as missing `<DeliveryType>:`) are ignored by the UMS Server for the purpose of driver selection.

The email client inconsistently receives notifications.	The Incoming Mail Server settings in the UMS Email Driver are configured with the same email account to which notifications are being sent. If the notification is sent to the same account, the UMS Email Driver may download and process the email before the email client can display it.	Use an exclusive e-mail account for Incoming Mail Server settings. Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `IncomingMailIDs` ■ `IncomingUserIDs`

Table 11–26 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions
SOA Human Workflow notifications are sent, but are not actionable.	The Actionable Email Address is not configured in SOA Workflow Notification Properties.	Set the Actionable Email Address in SOA Workflow Notification Properties with the address of the email account configured in the UMS Email Driver.
	The Human Workflow task is not set to send actionable notifications.	Set the *actionable* attribute for the Human Workflow task in JDeveloper and redeploy the SOA composite application.
SOA Human Workflow actionable notifications are sent, but no action is taken after responding.	The Incoming Mail Server settings in the UMS Email Driver are incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `MailAccessProtocol` (*IMAP* or *POP3*, in uppercase) ■ `ReceiveFolder` ■ `IncomingMailServer` ■ `IncomingMailServerPort` ■ `IncomingMailServerSSL` ■ `IncomingMailServerSSL` ■ `IncomingUserIDs` ■ `IncomingUserPasswords` ■ `ImapAuthPlainDisable` Note: Validate the values by using them in any e-mail client for connecting to an IMAP or POP3 server.
	The mail access protocol is incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `MailAccessProtocol` (*IMAP* or *POP3*, in uppercase)
	The email server is SSL-enabled.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `IncomingMailServerSS`
	The receive folder name is incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: ■ `ReceiveFolder` Note: Some email servers may expect the value INBOX to be inbox or Inbox (that is, case-sensitive). Based on your email server, use an appropriate value.
	A non-default email client is configured for receiving notifications. When the user clicks the approval link, the default mail client page opens, which may send emails to a different email server.	Configure the default email client to receive actionable notifications.

Table 11–26 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions
SOA BPEL User Notification or Human Workflow notifications are sent to the correct delivery type (email, sms, and so on) but to the wrong address.	A self-provisioned messaging channel was created by the user in User Messaging Preferences for use in BPEL User Notification or Human Workflow use cases. Note: The User Messaging Preferences UI allows the end user to create his or her own messaging channel for various use cases, but these are not to be used for BPEL User Notification and Human Workflow.	Do not use a self-provisioned messaging channel for BPEL User Notification or Human Workflow use cases (that is, do not set as Default channel, and do not use in a messaging filter for such use cases). BPEL User Notification and Human Workflow utilize User Messaging Preferences only for the delivery type preference, and the actual address is retrieved from the user profile in the identity management system. Note: Addresses from the user profile in the identity management system are available through User Messaging Preferences using pre-defined channel names, such as *Business Email*, *Business Mobile*, *Business Phone*, *Instant Messaging*. Use these pre-defined messaging channels instead for BPEL User Notification and Human Workflow use cases.

12

Monitoring Oracle User Messaging Service

This chapter describes how to monitor Oracle User Messaging Service using Oracle Enterprise Manager Fusion Middleware Control.

This chapter includes the following topics:

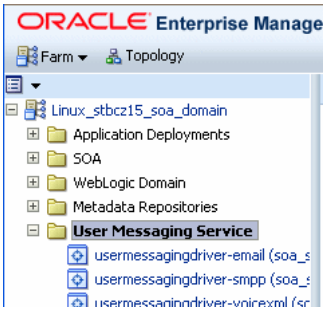
- Section 12.1, "Monitoring Oracle User Messaging Service"
- Section 12.2, "Log Files"
- Section 12.3, "Metrics and Statistics"

12.1 Monitoring Oracle User Messaging Service

You can monitor Oracle User Messaging Service logs and metrics using Oracle Enterprise Manager Fusion Middleware Control. To access this functionality:

1. Go to the Enterprise Manager page for your SOA farm.

Figure 12–1 Managing your SOA farm

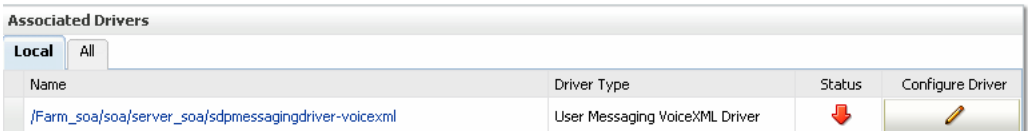


2. Select Fusion Middleware, SOA, User Messaging Service.
3. Select the server or driver of your choice.

   If you select a driver, you will see some quick statistics that indicate the state and performance of the driver.

   If you select a server, you will see a list of associated drivers, in addition to the quick statistics. You can select one of the drivers to view its statistics, or you can click the Configure Driver icon to configure it. For more information on configuring drivers, see Chapter 11, "Configuring Oracle User Messaging Service".

Figure 12–2 Using the Configure Driver icon



4. Right-click on a driver to take the actions listed in.

Figure 12–3 Available actions

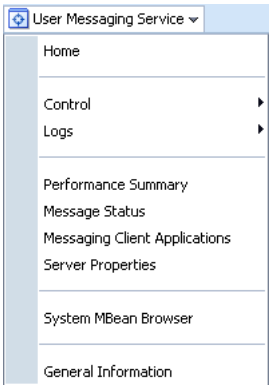
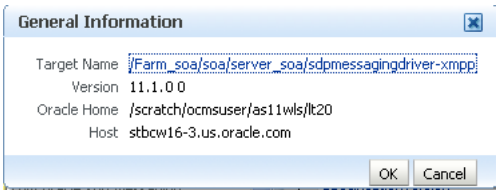


Table 12–1 Driver actions

Selection	Action
Home	The Home page lists the quick statistics for the selected driver.
Control	Start Up or Shut Down driver.
Logs	View and configure message logs for the selected driver.
Performance Summary	Displays Performance Statistics on a customizable metrics page. Use this page to view statistics for this driver. Customize this page using the Metric Palette. The Metric Palette enables you to choose from all of the available metrics so that you see only the information that is most valuable to you.
Message Status	Check the delivery status of messages sent and received, and resend selected messages. You can filter the search by adding more search fields and setting the desired operator and search value. Some fields can be added multiple times in order to use them with different and complementary operators, or with the *Contains* operator.
Messaging Client Applications	Messaging client applications registered with the User Messaging Service can be manually de-registered in cases where the applications have been undeployed and are holding onto access points that need to be made available to other applications.
Server Properties	Configure message storage method and business terms for message filter creation. See Chapter 11, "Configuring Oracle User Messaging Service" for more information.
System MBean Browser	System MBean Browser the System MBeans and their configuration settings.

Table 12–1 (Cont.) Driver actions

Selection	Action
General Information	General Information displays the name, version, Oracle Home, and host for the selected driver.



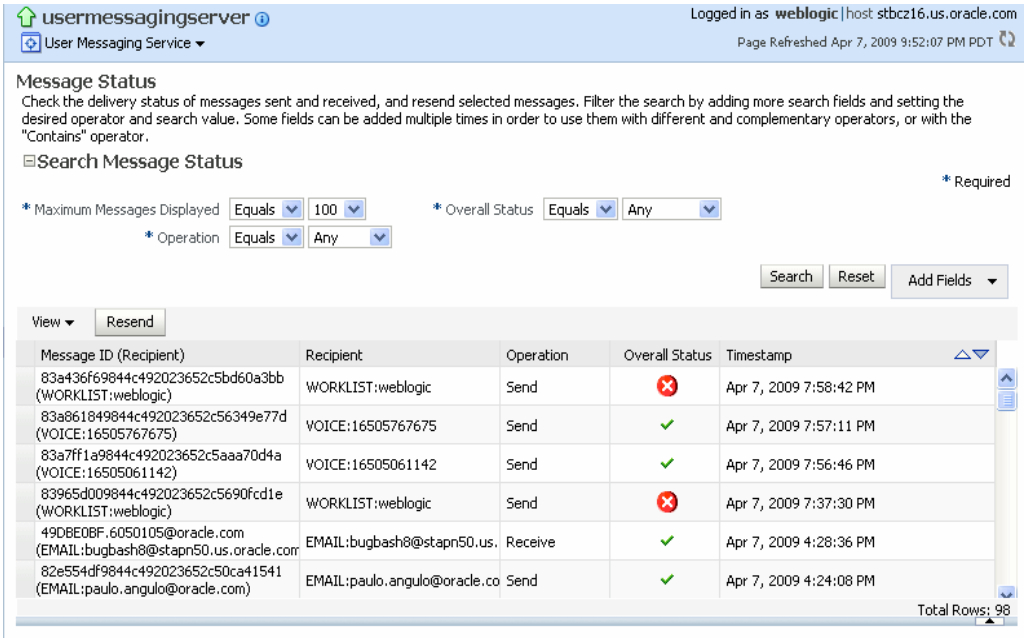
For more information about Oracle Enterprise Manager, see your Oracle Enterprise Manager documentation.

12.1.1 Using Message Status

You can check the delivery status of messages sent and received, and resend selected messages. To check message status:

1. In the navigation tree, right-click on the UMS target for which you want to view message status.
2. Select **Message Status**. The *Message Status* page appears.
3. Click **Search** to search the messages using the default criteria. The search returns a listing for the messages.

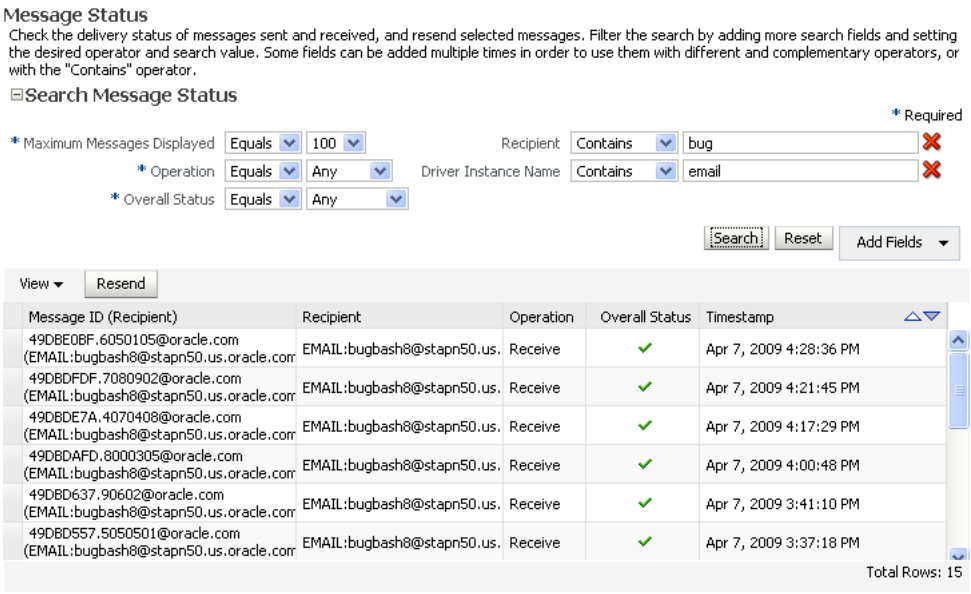
Figure 12–4 Message Status



You can customize the search by adding more search fields and setting the desired operator and search value. Some fields can be added multiple times in order to use them with different and complementary operators, or with the Contains operator. To customize the search:

1. Click **Add Fields**.
2. Select the field(s) on which you want to search.
3. Choose operators and fill in variables as needed.
4. Click **Search**. The customized search is done and results returned.

Figure 12–5 Custom search



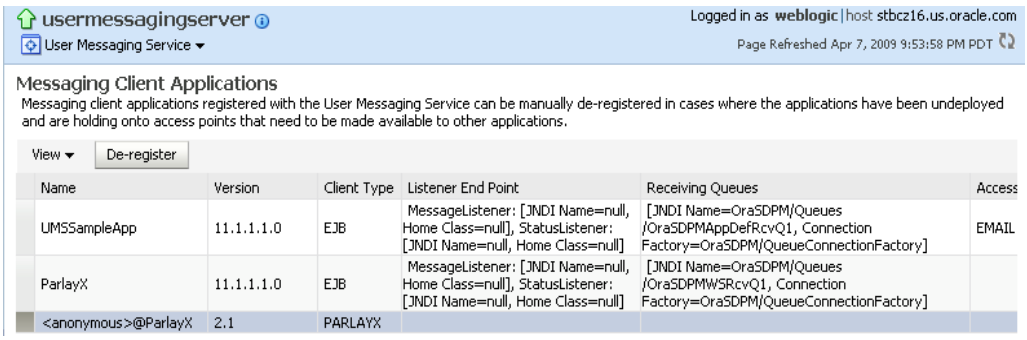
5. If you want to resend a message, select the message in the list and click **Resend**.

12.1.2 Deregistering Messaging Client Applications

You can manually deregister Messaging Client Applications after the applications have been undeployed and are holding onto access points that need to be made available to other applications. To deregister Messaging Client Applications:

1. Right-click a target in the navigation tree, and select **Messaging Client**. The Messaging Client page appears.
2. Select the message that you want to deregister.
3. Click **De-register**.

Figure 12–6 Messaging Client Applications page



A confirmation box appears asking you to confirm your choice.

4. Confirm your choice.

12.1.3 Monitoring Drivers Using the All Tab

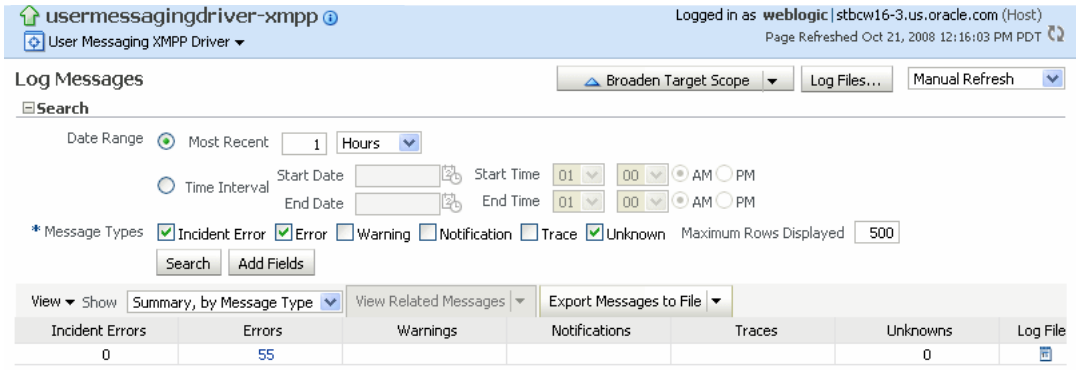
The **All** tab only lists successfully-registered drivers in the domain (not all drivers that exist).

Since the drivers are not configured out-of-the-box, they are not registered unless you configure them. To ensure that you see all of the drivers in the v tab, configure the SMPP, VoiceXML and XMPP drivers (if you plan to use them). Once configured, they will register with the engine and they will be displayed in the **All** tab.

12.2 Log Files

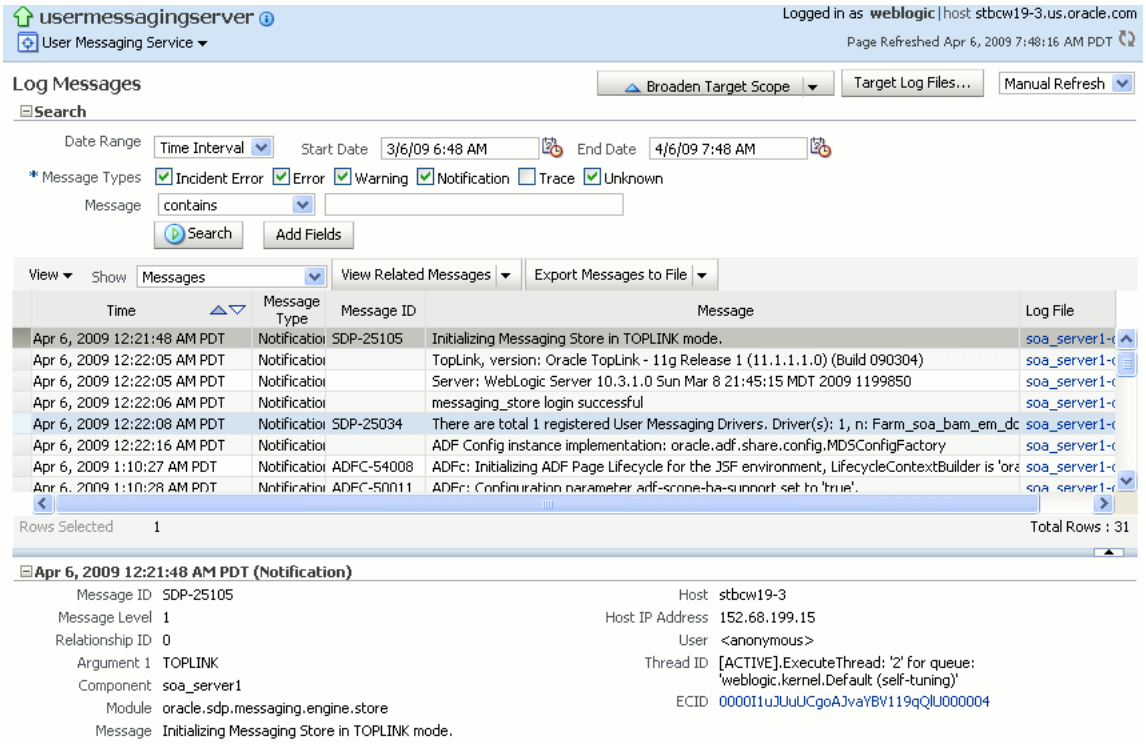
Right-click on the driver for which you want to view log information, the choose Logs, View Log Files. The Log Messages page appears.

Figure 12–7 Querying logs



Use this page to query for log information about the driver. Fields and lists are used to customize the query. After entering your search criteria, click Log Files. The Log Files page appears.

Figure 12–8 Log search results

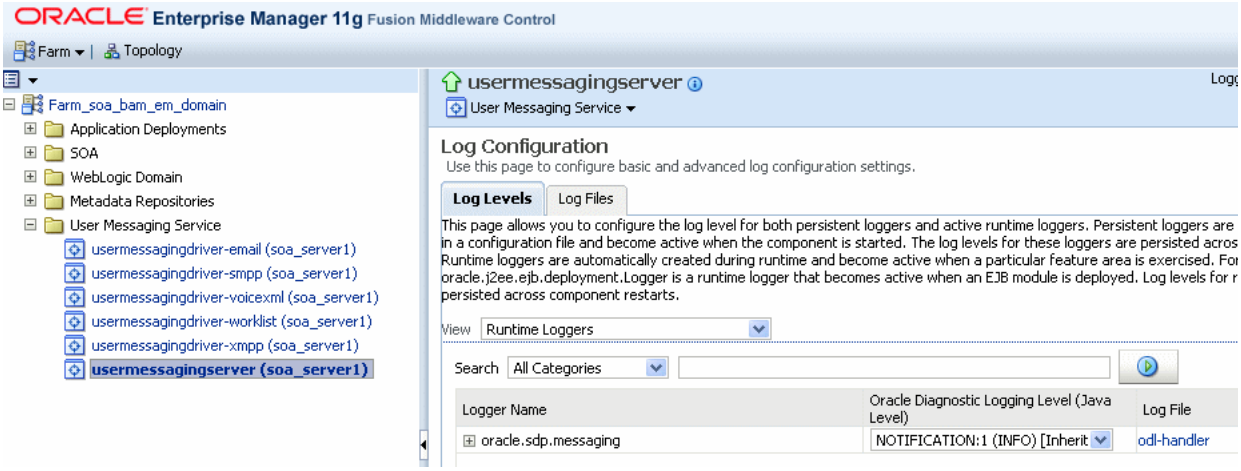


You can view log information or download the log.

12.2.1 Configuring Logging

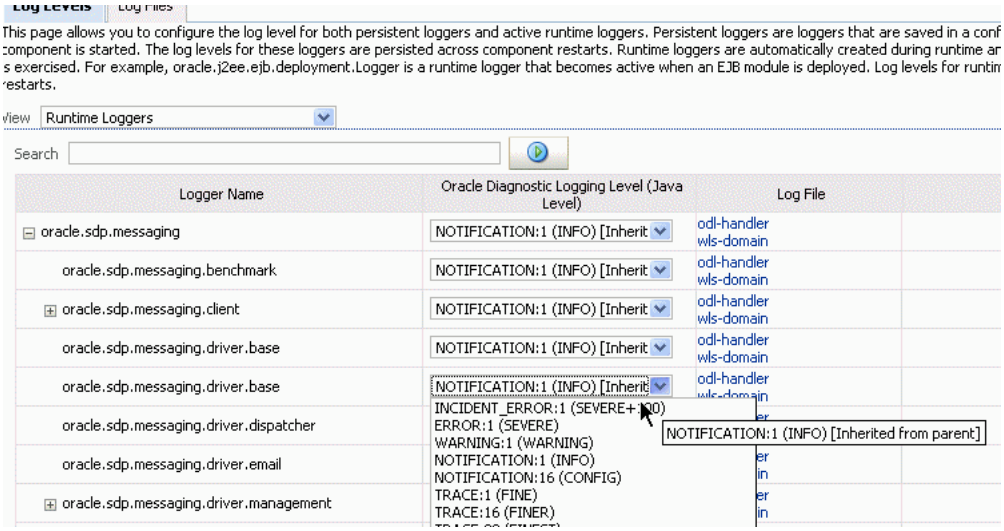
Use Enterprise Manager to configure log levels.

Figure 12–9 Configuring log levels



For each logger, set the notification level.

Figure 12–10 Select notification level



As a result of your configuration actions, notifications will appear according to your specification.

Figure 12–11 Viewing log files

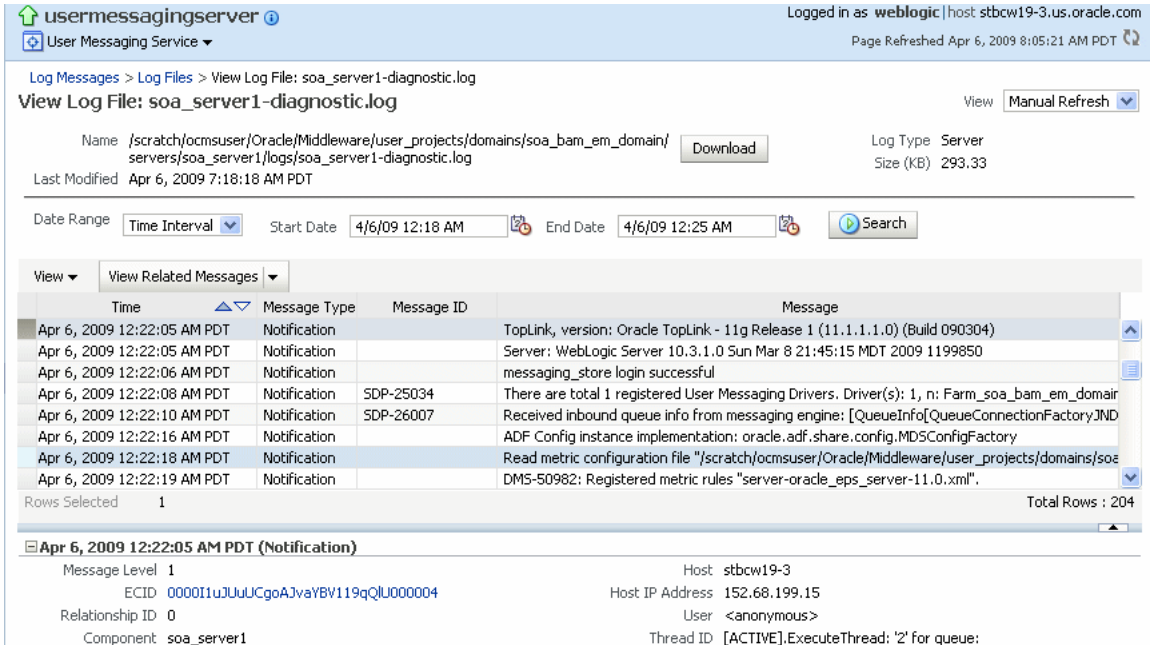
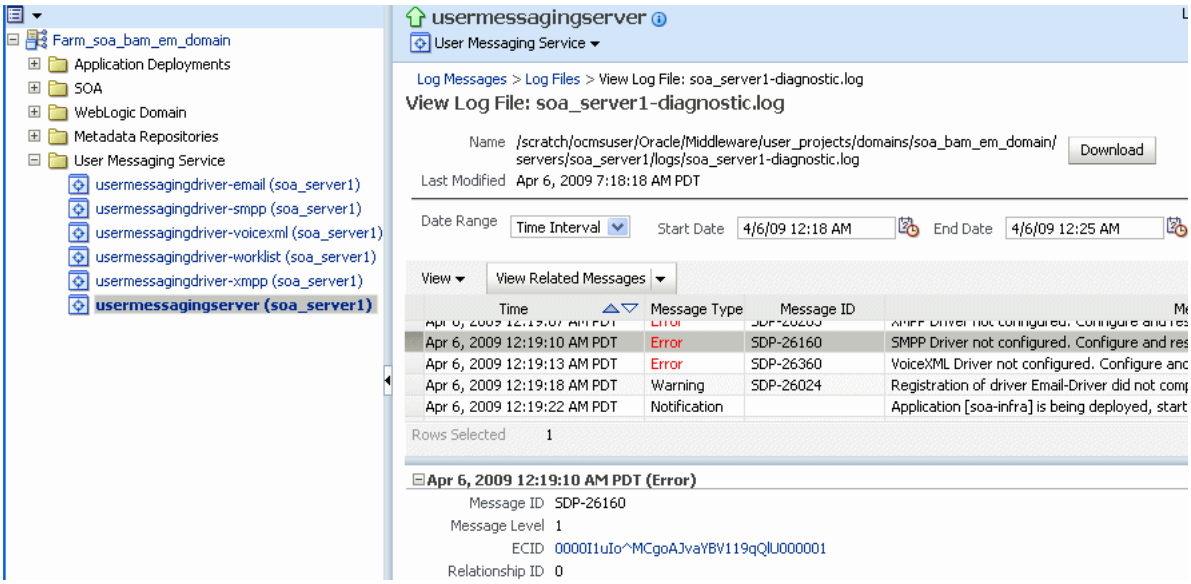


Figure 12–12 Error messages



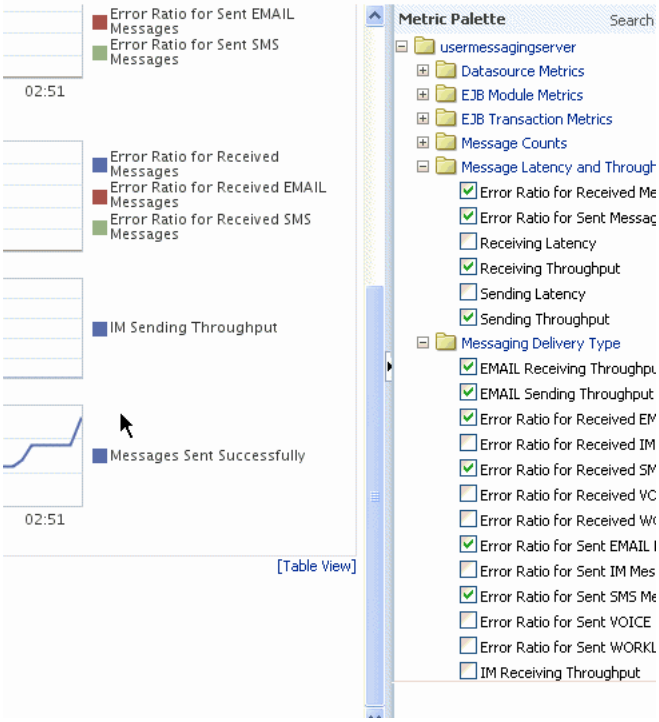
12.3 Metrics and Statistics

The performance of your applications is reflected in metrics and statistics. When you select the Performance Summary for a driver, the Performance Summary page appears.

Figure 12–13 UMS metrics

Many metrics are available for capture and display, but in order to get the most valuable, focused information, use Metric Palette. Click Show Metric Palette to display the Metric Palette. Choose the metrics in which you are most interested. As you select or deselect metrics from the palette, the metrics display is automatically updated.

Figure 12–14 Metrics Palette



13

Managing Oracle User Messaging Service

This chapter describes how to manage Oracle User Messaging Service.

This chapter includes the following topic:

- Section 13.1, "Deploying Drivers"
- Section 13.2, "Undeploying and Unregistering Drivers"

13.1 Deploying Drivers

When you install Oracle UMS, pre-installed drivers are included (Email, XMPP, SMPP, and VoiceXML). Of these, only the Email driver is deployed to the WebLogic Server. To deploy the others, target that driver to the WebLogic Server (using WebLogic Administration Console, or you can target the drivers when creating or extending the domain using the Oracle Fusion Middleware Configuration Wizard).

The Worklist driver must be deployed to a SOA Server if you want to make use of the UMS integration with Worklist. Because this integration involves multiple JEE applications and a SOA composite, there is a special extension template you must use to enable this feature in one step. See Install the Worklist Driver for more information.

You can deploy additional drivers in a variety of ways using: WebLogic Server Administration Console, Oracle Enterprise Manager, WLST commands, and through the Oracle Fusion Middleware Configuration Wizard.

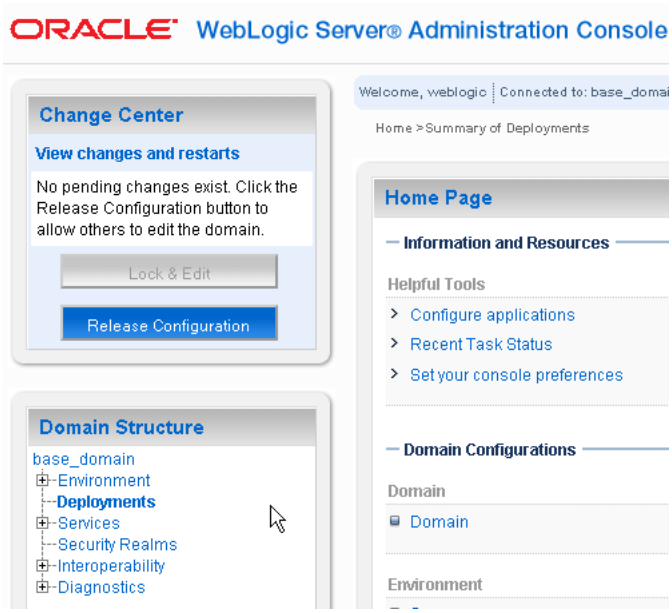
> **Note:** To deploy two or more driver instances of a particular driver EAR, you must use the custom deployment plan templates available at `$ORACLE_HOME/communications/plans`. See Using Oracle Enterprise Manager to Deploy Drivers for instructions on deploying drivers using Oracle Enterprise Manager.

13.1.1 Using WebLogic Server Administration Console

Use WebLogic Server Administration Console to deploy drivers.

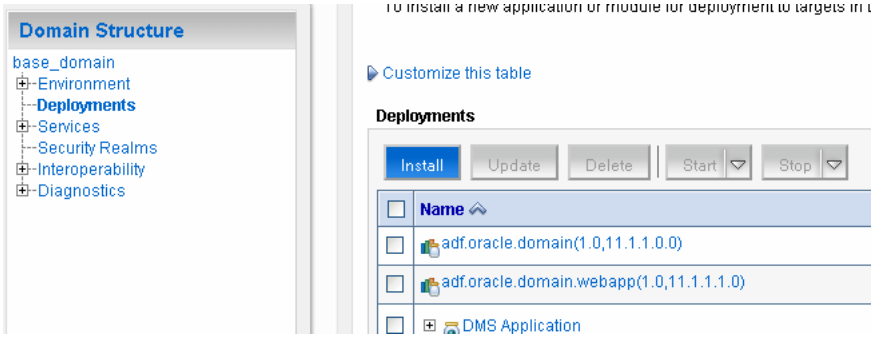
1. In the Domain Structure region of the console, click Deployments. The Home page for Deployments appears.

Figure 13–1 Deployments



2. Under Deployments, click Install.

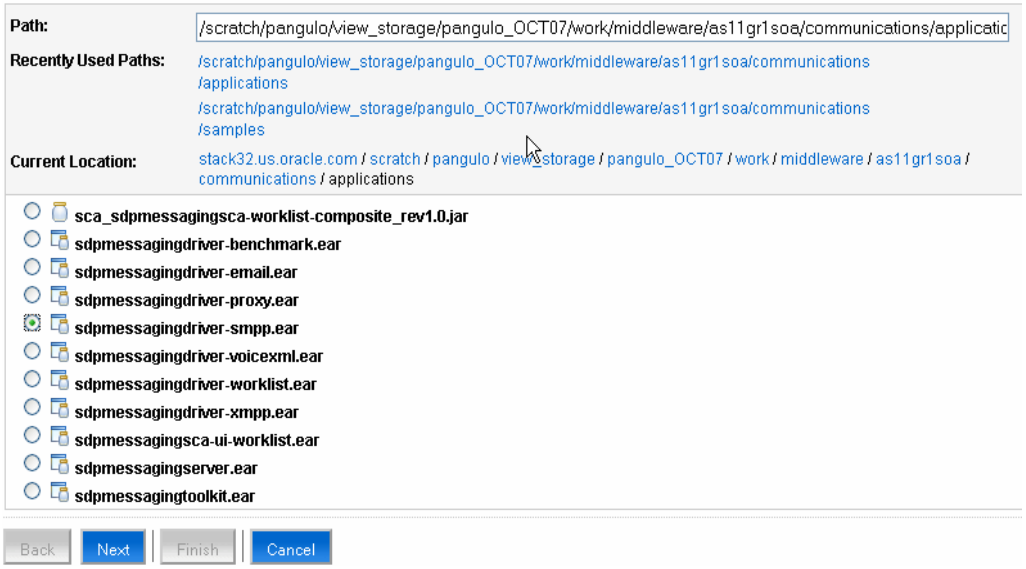
Figure 13–2 Install



The Install Application Assistant appears. Use this page to locate the application you want to deploy.

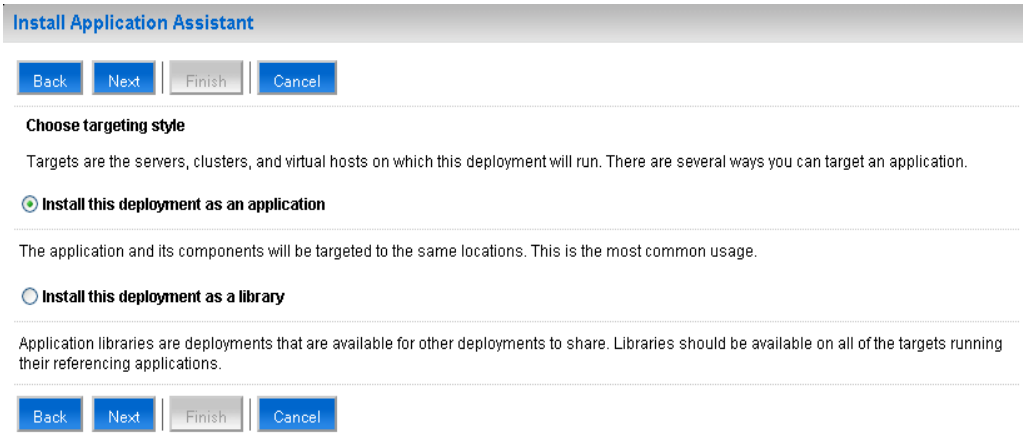
3. Enter the path to your file.

Figure 13–3 Install Application Assistant



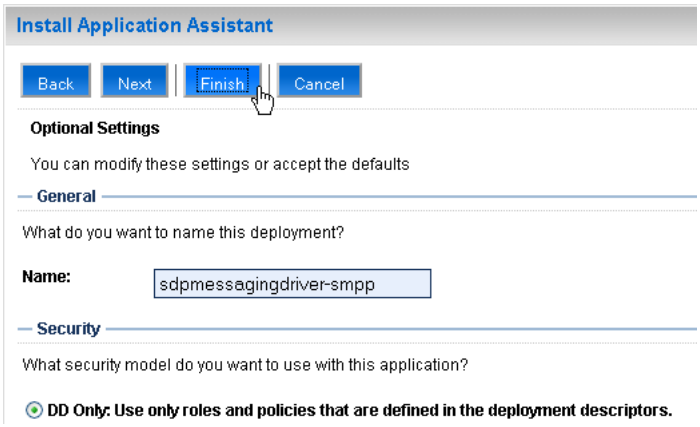
4. Click Next. You will be asked to choose the targeting style.

Figure 13–4 Targeting style



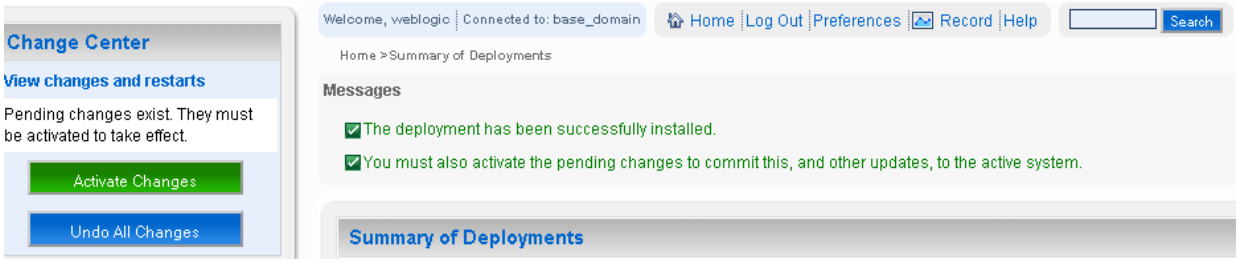
5. Use the Default (Install this deployment as an application). A Summary page appears.

Figure 13–5 Summary page



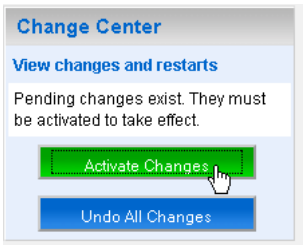
6. Accept the settings. You can change setting here, but it is recommended that you accept the settings as they are. Click Finish. A Confirmation page appears.

Figure 13–6 Confirmation page



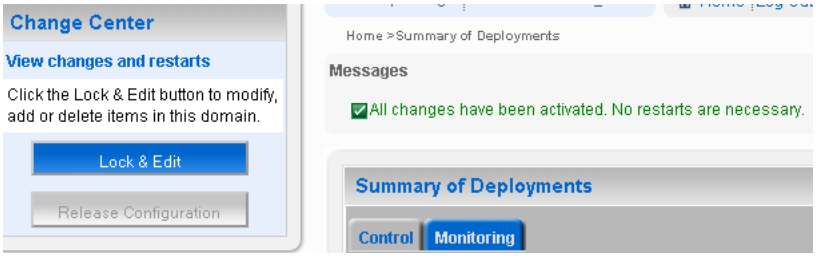
7. In order for your deployment to be complete, you must activate your changes, so click Activate Changes.

Figure 13–7 Activate changes



A final confirmation appears.

Figure 13–8 Final confirmation



13.1.2 Using Oracle Enterprise Manager to Deploy Drivers

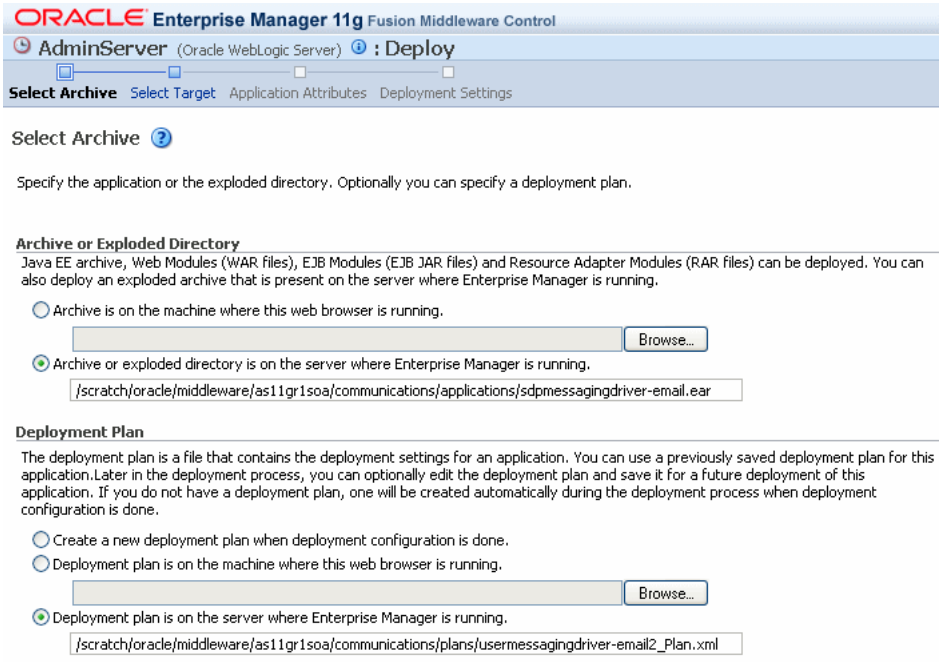
Follow these steps to deploy drivers using Oracle Enterprise Manager.

1. Retrieve a deployment template (for example: `ORACLE_HOME/communications/plans`)
2. Copy the plan to a location of your choice (to the same directory or any other directory).
3. Edit the plan:

   Replace *DriverDeploymentName* with whichever name you want to use (ensure you replace all instances of the name).

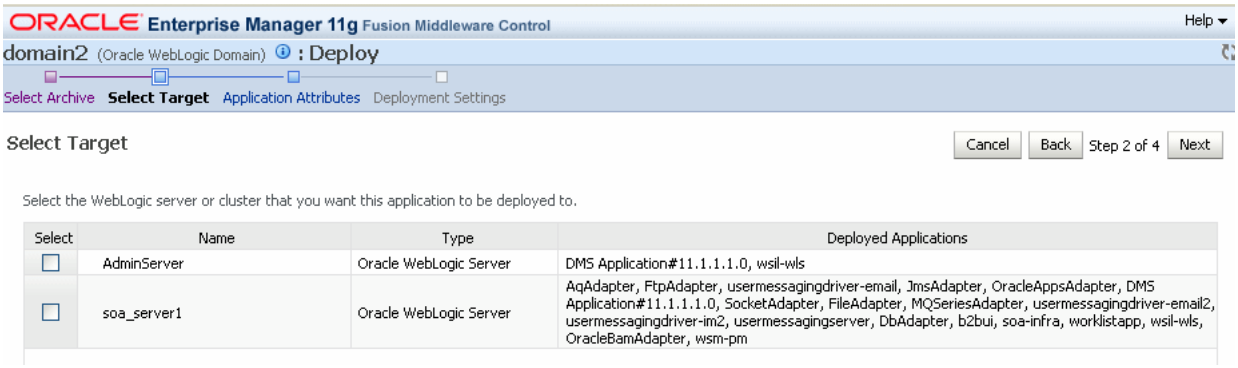
   Replace DriverShortName with any name you like.
4. Start Oracle Enterprise Manager.
5. Enter the location of the .ear file (Figure 13–9).
6. Enter the location of the Deployment Plan (Figure 13–9).

Figure 13–9 Deploying UMS Drivers using Oracle Enterprise Manager



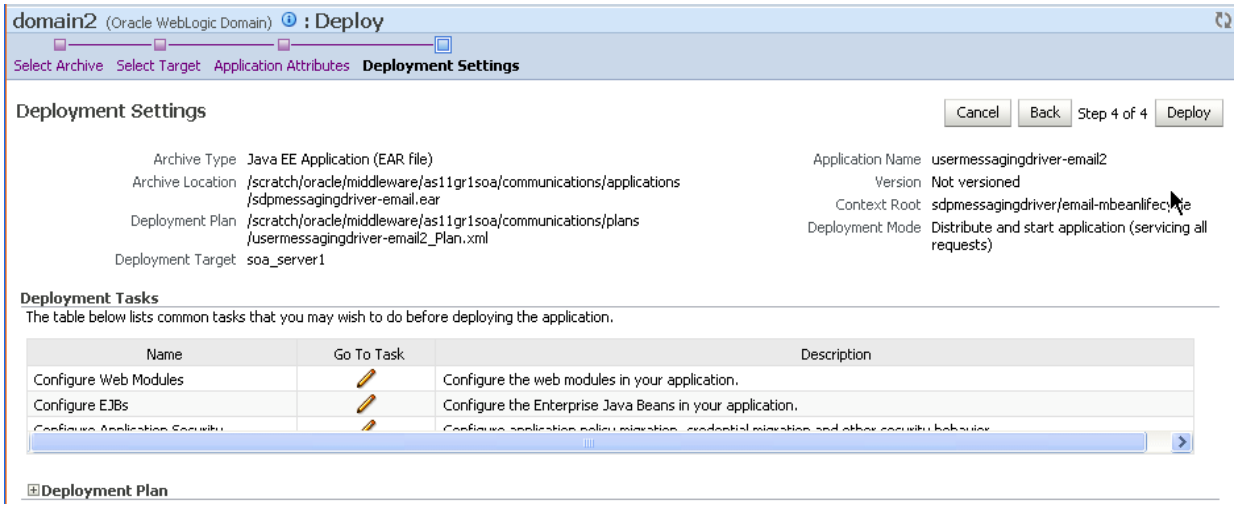
The Select Target screen appears.

Figure 13–10 Select Target screen



7. Select the SOA target.
8. Enter an application name in the Application Attributes screen. The application name must exactly match the string used for *DriverDeploymentName* (in Step 3 above) which is provided in the Deployment Plan. If it does not, the deployment and activation will fail. The Deployment Setting screen appears.

Figure 13–11 Deployment Settings screen



9. Click **Deploy**. The Deployment Completed screen appears.

Figure 13–12 Deployment Completed screen

10. To see the result (driver deployed), start the SOA Server.

13.1.3 Using WLST Commands

You can deploy drivers using the WLST command `deployUserMessagingDriver`.

13.1.3.1 deployUserMessagingDriver

Command Category: UMS

Use with WLST: Online

13.1.3.1.1 Description `deployUserMessagingDriver` is used to deploy additional instances of user messaging drivers.

Specify a base driver type (for example: email, xmpp, voicexml, and others) and a short name for the new driver deployment. The string *usermessagingdriver-* will be prepended to the specified application name. Any valid parameters for the *deploy* command can be specified, and will be passed through when the driver is deployed.

13.1.3.1.2 Syntax `deployUserMessagingDriver(baseDriver, appName, [targets], [stageMode], [options])`

Argument	Definition
`baseDriver`	Specifies the base messaging driver type. Must be a known driver type, such as 'email', 'proxy', 'smpp', 'voicexml', or 'xmpp'.
`appName`	A short descriptive name for the new deployment. The specified value will be prepended with the string *usermessagingdriver-*

Argument	Definition

`targets`
`stageMode`
`options`
	Optional. Additional arguments that are valid for the *deploy* command can be specified and will be passed through when the new driver is deployed.

13.1.3.1.3 Examples To deploy a second instance of an email driver with name *myEmail*.

```
wls:/base_domain/serverConfig> deployUserMessagingDriver(baseDriver='email',
appName='myEmail')
```

To deploy a second instance of an email driver, specifying deployment targets.

```
wls:/base_domain/serverConfig> deployUserMessagingDriver(baseDriver='email',
appName='email2', targets='server1,server2')
```

13.1.4 Using the Oracle Fusion Middleware Configuration Wizard

To install the SMPP, XMPP and VoiceXML drivers, extend the domain using the extension template available at `$ORACLE_HOME/common/templates/applications/oracle.ums.drivers_template_11.1.1.jar`.

To extend a domain using Oracle Fusion Middleware Configuration Wizard:

1. Launch Oracle Fusion Middleware Configuration Wizard (`$ORACLE_HOME/common/bin/config.sh or %ORACLE_HOME%\common\bin\config.cmd`).
2. Select the *Extend an existing WebLogic domain* option.
3. Select the desired domain directory containing UMS.
4. Select the *Extend my domain using an existing extension template* option.
5. Click **Browse**, and navigate to `$ORACLE_HOME/common/templates/applications`
6. Select `oracle.ums.drivers_template_11.1.1.jar`
7. Complete the remaining steps of the Oracle Fusion Middleware Configuration Wizard, and remember to target the required drivers to the desired WebLogic servers and/or clusters.
8. Restart the appropriate WebLogic servers.

13.2 Undeploying and Unregistering Drivers

Since Messaging Drivers are standard JEE applications, they can be undeployed from the Oracle WebLogic Server using standard Oracle WebLogic tools such as the Admin Console or WLST.

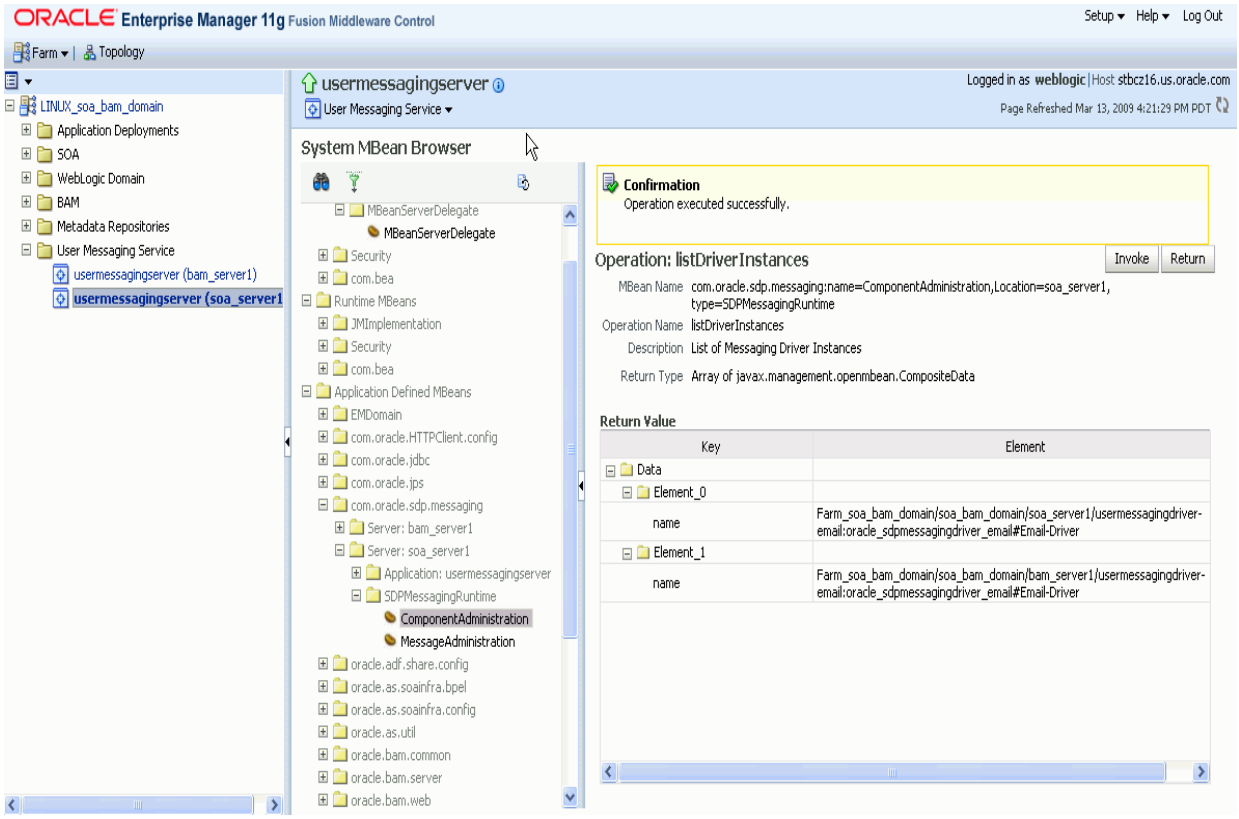
However, since the UMS server keeps track of the messaging drivers that have been registered with it in a persistent store (database), this registration must be cleaned in a separate step using a runtime MBean exposed by the UMS server. The procedure to do this from Oracle Enterprise Manager is as follows:

1. Ensure the UMS server is available.

2. In Oracle Enterprise Manager, select any `usermessagingserver` target in the domain.
3. From the target's menu, select *System MBean Browser*.
4. In System MBean Browser, locate the *ComponentAdministration* MBean of `usermessagingserver`:

   Expand the folder *com.oracle.sdp.messaging* > **Server** (such as *Server: soa_server1*) > **SDPMessagingRuntime** > **ComponentAdministration**.
5. Invoke the operation *listDriverInstances*.
   a. Click the **Operations** tab.
   b. Click the operation *listDriverInstances*.
   c. Click **Invoke**.
   d. Identify and copy the name of the driver you want to unregister. (for example: `/Farm_soa_bam_domain/soa_bam_domain/soa_server1/usermessagingdriver-email:oracle_sdpmessagingdriver_email#Email-Driver`)

Figure 13–13 Listing Driver Instances

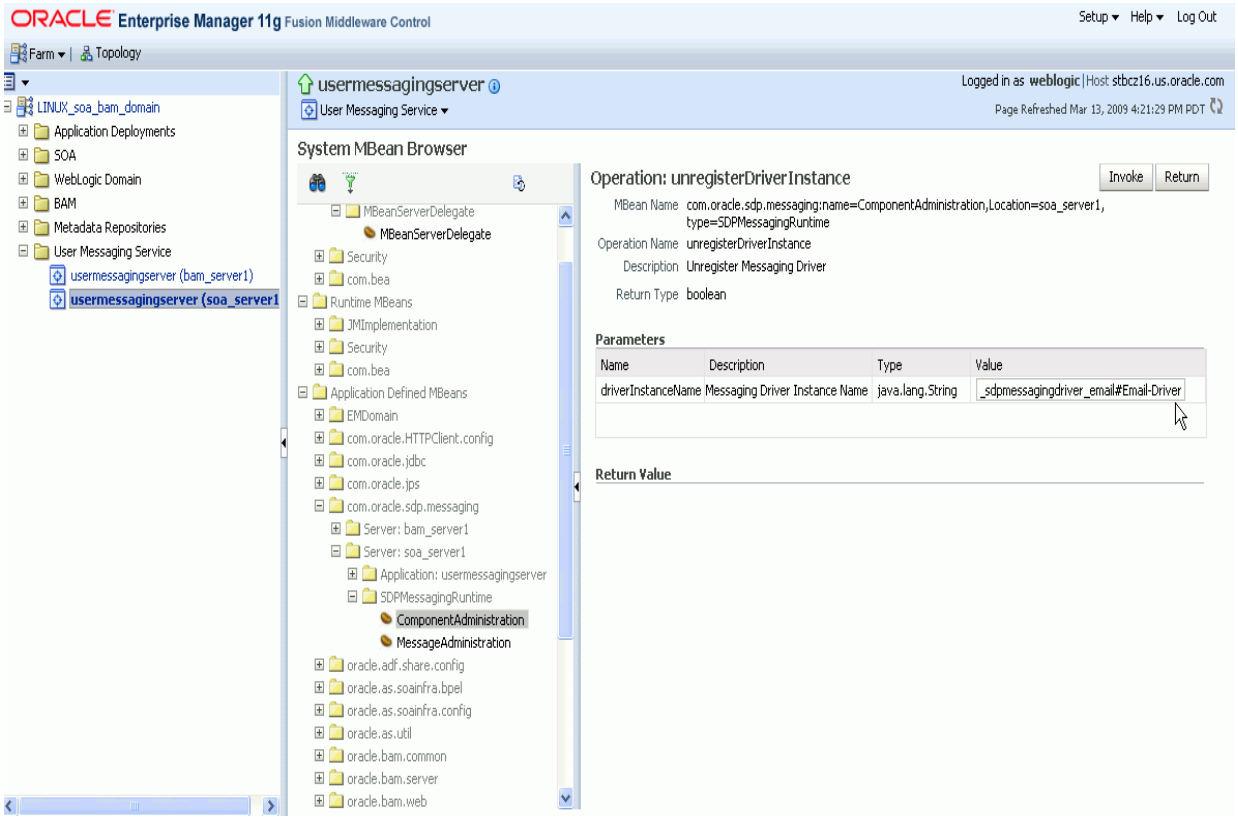


6. Click **Return**.
7. Invoke the operation *unregisterDriverInstance* with the desired driver name.
   a. Click the operation *unregisterDriverInstance*.
   b. Paste the driver name in the *Value* field (for example: `/Farm_soa_bam_domain/soa_bam_domain/soa_`

`server1/usermessagingdriver-email:oracle_sdpmessagingdriver_email#Email-Driver`).

c. Click **Invoke**.

Figure 13–14 Unregistering a Driver Instance



8. Check the confirmation dialog for success.

This completes the unregistration of the specified driver from the UMS server and it will no longer be used in future message delivery.

Part V

Confiding SIP Infrastructure Applications

This Part contains the following chapter:

- Chapter 15, "Oracle WebLogic Communication Services Base Platform Topologies"

14

Configuring SIP Infrastructure Applications

This chapter describes Proxy Registrar and STUN Service. Topics include:

- Section 14.1, "Proxy Registrar"
- Section 14.2, "STUN Service"

14.1 Proxy Registrar

The Proxy Registrar is a user agent server (UAS) that implements the proxy and registrar functions described in RFC 3261. This SIP entity is a router of messages. The Proxy Registrar's registrar function processes the REGISTER requests from User Agent clients and uses a Location Service to store a binding (that is, an association) between a user's address of record (AOR) and the user's SIP or SIPS URIs that are located in a CONTACT field. Upon receiving requests to the AOR, the proxy function locates the mapped URIs through a Location Service lookup and then proxies the request using the location information retrieved by this lookup. Table 14–1 describes the attributes of the Proxy Registrar.

Table 14–1 Attributes of the Proxy Registrar

Attributes	Description
CurrentRegDevices	A read-only attribute that displays the number of currently registered devices.
DefaultExpires	Sets the expiration value for the REGISTER request if the client has not indicated a preferred value itself. The default value for this attribute is 3600 seconds.
MaxExpires	Sets the maximum expiration value for the REGISTER request accepted by the server. Although a client can request any expiration value in the REGISTER request, the server can set a maximum amount of time that it accepts for expiration. If the client requests a time greater than the value set for *MaxEpires*, then the server sets the expiration time for that particular REGISTER request to the value set for *MaxExpires*. The default value for this attribute is 7200 seconds.

Table 14–1 (Cont.) Attributes of the Proxy Registrar

Attributes	Description
MinExpires	Specifies the minimum expiration value for a REGISTER request accepted by server. While clients can request any expiration time, they can also specify a very low value for the expiration of the REGISTER request. Such low values require clients to update registration information frequently, which creates traffic on the network. If a client requests a value that is below this minimum expiration time, then the server does not accept the REGISTER request and responds with a 423 (*Interval Too Brief*) error response per RFC 3261. This response message specifies the lowest expiration time allowed, which is set by the *MinExpires* attribute. The server is allowed to shorten an expiration time, but can never lengthen one. The default value for this attribute is 60 seconds.
SipRegAllowThirdParty	Specifies whether the Proxy Registrar allows third-party registrations. In a third-party registration, the entity issuing the request (in the *From* header) is different from the entity being registered (in the *To* header) to whom the provided Contact information applies. If set to true, the Proxy Registrar allows third party registrations. If set to false (the default value), then third-party registrations are rejected (the requestor receives a *403 Forbidden* status code). This is a read-only attribute that is always set to *false*.
SipRegMaxUsers	A read-only attribute that specifies the maximum number of users supported by the Proxy Registrar.

14.2 STUN Service

The OWLCS STUN Service implements STUN -- Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). As described in RFC 3489, STUN enables STUN clients behind a NAT (that is, clients behind a router) to discover the presence of a NAT, the type of NAT, and then to learn the address bindings (including IP addresses) allocated by the NAT.

STUN is a client-server protocol in which a STUN client sends a request (a Binding Request) to a server, which in turn sends a response. OWLCS supports the receipt of Binding Requests from a client, which are sent over UDP and are used to both discover the presence of a NAT and discover the public IP address and the port mappings that it generates. When a STUN client sends a Binding Request to the STUN server, the STUN Server examines the request's source IP address and port and copies them into a response that it sends back to the client. When the STUN client receives the Binding Response, it compares the IP address and port in the packet with the local IP address and port to which it bound itself when it sent the Binding Request to the STUN Server.

The attributes of the STUN Service MBean (described in Table 14–2) enable you to set the STUN Server's primary and secondary IP addresses and ports that form the four RFC 3489-dictated address-port combinations used by the STUN server to receive client Binding Requests. Per RFC 3489, the combinations are as follows:

- A1, P1 -- The Primary Address and Primary Port
- A2, P1 -- The Secondary Address and the Primary Port
- A1, P2 -- The Primary Address and the Secondary Port
- A2, P2 -- The Secondary Address and the Secondary Port

Typically, the STUN server's Primary Port (P1) is set to UDP port 3478. The Stun server uses the Secondary Address and Secondary Port values (A2, P2) in the `CHANGED-ADDRESS` attribute included in its Binding Response.

Table 14–2 Attributes of the STUNService MBean

Attribute	Value
Autostart	Set to *true* for the Stun Server to start automatically when OWLCS starts.
PrimaryAddress	The primary STUN address on which to listen for incoming Binding Requests. The default value is 127.0.0.1.
PrimaryPort	The primary STUN port on which to listen for incoming Binding Requests. The value is UDP port 3478, the default STUN Port as described in RFC 3489.
SecondaryAddress	The secondary STUN address on which to listen for incoming Binding Requests. This cannot be the same value as *PrimaryAddress*.
SecondaryPort	The secondary STUN port to which to listen for incoming Binding Requests. The default value is UDP port 3479.

Part VI

Deploying Oracle WebLogic Communication Services

This Part contains the following chapters:

15

Oracle WebLogic Communication Services Base Platform Topologies

The following sections provide an overview of the Oracle WebLogic Communication Services architecture and deployment topologies:

- Section 15.1, "Goals of the Oracle WebLogic Communication Services Base Platform"
- Section 15.2, "Load Balancer"
- Section 15.3, "Engine Tier"
- Section 15.4, "SIP Data tier"
- Section 15.5, "Geographically-Redundant Installations"
- Section 15.6, "Example Hardware Configurations"
- Section 15.7, "Alternate Configurations"

15.1 Goals of the Oracle WebLogic Communication Services Base Platform

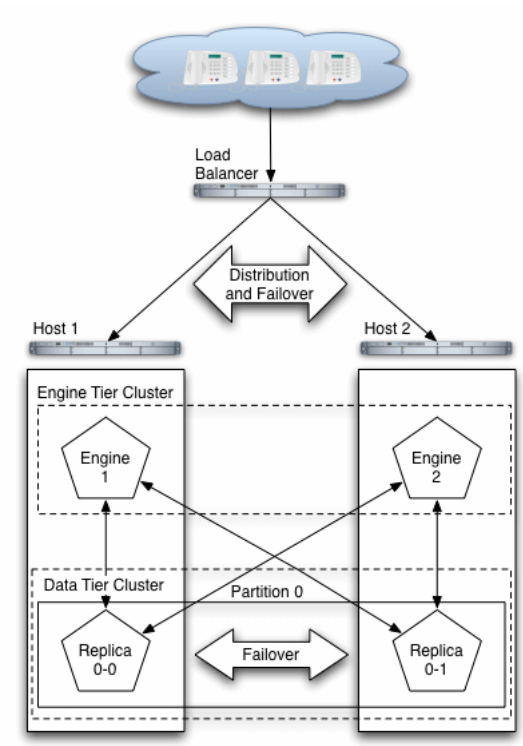
Oracle WebLogic Communication Services is designed to provide a highly scalable, highly available, performant server for deploying SIP applications. OWLCS is used for deploying any JSR 289 compliant SIP applications. For supported topologies for all the services in OWLCS, see Chapter 16, "Deployment Topologies for Communication Services".

The Oracle WebLogic Communication Services architecture is simple to manage and easily adaptable to make use of available hardware. The basic architecture consists of these components:

- Section 15.2, "Load Balancer"
- Section 15.3, "Engine Tier"
- Section 15.4, "SIP Data tier"

Figure 15–1 shows the components of a basic Oracle WebLogic Communication Services installation. The sections that follow describe each component of the architecture in more detail.

Figure 15–1 Oracle WebLogic Communication Services Architecture



15.2 Load Balancer

Although it is not provided as part of the Oracle WebLogic Communication Services product, a load balancer (or multiple load balancers) is an essential component of any production Oracle WebLogic Communication Services installation. The primary goal of a load balancer is to provide a single public address that distributes incoming SIP requests to available servers in the Oracle WebLogic Communication Services engine tier. Distribution of requests ensures that Oracle WebLogic Communication Services engines are fully utilized.

Most load balancers have configurable policies to ensure that client requests are distributed according to the capacity and availability of individual machines, or according to any other load policies required by your installation. Some load balancers provide additional features for managing SIP network traffic, such as support for routing policies based on source IP address, port number, or other fields available in SIP message headers. Many load balancer products also provide additional fault tolerance features for telephony networks, and can be configured to consistently route SIP requests for a given call to the same engine server on which the call was initiated.

In a Oracle WebLogic Communication Services installation, the load balancer is also essential for performing maintenance activities such as upgrading individual servers (Oracle WebLogic Communication Services software or hardware) or upgrading applications without disrupting existing SIP clients. The Administrator modifies load balancer policies to move client traffic off of one or more servers, and then performs the required upgrades on the unused server instances. Afterwards, the Administrator modifies the load balancer policies to allow client traffic to resume on the upgraded servers.

Oracle provides detailed information for setting up load balancers with the Oracle WebLogic Communication Services engine tier for basic load distribution. See

Section 4.2, "Configuring Load Balancer Addresses" to configure a load balancer used with Oracle WebLogic Communication Services.

15.3 Engine Tier

The engine tier is a cluster of Oracle WebLogic Communication Services instances that hosts the SIP Servlets and other applications that provide features to SIP clients. The engine tier is a stateless cluster of servers, and it stores no permanent or transient information about the state of SIP dialogs. Instead, all stateful information about SIP dialogs is stored and retrieved from SIP Data Tier (Section 15.4), which also provides replication and failover services for SIP session data.

Engine tier servers can optionally cache a portion of the session data managed by the SIP data tier. Caching is most useful in configurations that use a SIP-aware load balancer. See Section 6.7, "Caching SIP Data in the Engine Tier".

The primary goal of the engine tier is to provide maximum throughput and low response time to SIP clients. As the number of calls, or the average duration of calls to your system increases, you can easily add additional server instances to the engine tier to manage the additional load.

Note that although the engine tier consists of multiple Oracle WebLogic Communication Services instances, you manage the engine tier as a single, logical entity; SIP Servlets are deployed uniformly to all server instances (by targeting the cluster itself) and the load balancer need not maintain an affinity between SIP clients and servers in the engine tier.

Note: Oracle WebLogic Communication Services start scripts use default values for many JVM parameters that affect performance. For example, JVM garbage collection and heap size parameters may be omitted, or may use values that are acceptable only for evaluation or development purposes. In a production system, you must rigorously profile your applications with different heap size and garbage collection settings in order to realize adequate performance. See Section 8.8, "Tuning JVM Garbage Collection for Production Deployments" for suggestions about maximizing JVM performance in a production domain.

Because the engine tier relies on SIP data tier servers in order to retrieve call state data, Oracle recommends using dual, Gigabit Ethernet Network Interface Cards (NICs) on engine and SIP data tier machines to provide redundant network connections.

15.4 SIP Data tier

The SIP data tier is a cluster of Oracle WebLogic Communication Services instances that provides a high-performance, highly-available, in-memory database for storing and retrieving the session state data for SIP Servlets. The goals of the SIP data tier are as follows:

- To provide reliable, performant storage for session data required by SIP applications in the Oracle WebLogic Communication Services engine tier.
- To enable administrators to easily scale hardware and software resources as necessary to accommodate the session state for all concurrent calls.

Within the SIP data tier, session data is managed in one or more "partitions" where each partition manages a fixed portion of the concurrent call state. For example, in a system that uses two partitions, the first partition manages one half of the concurrent call state (sessions A through M) while the second partition manages another half of the concurrent call states (sessions N through Z). With three partitions, each partition manages a third of the call state, and so on. Additional partitions can be added as necessary to manage a large number of concurrent calls. A simple hashing algorithm is used to ensure that each call state is uniquely assigned to only one SIP data tier partition.

Within each partition, multiple servers can be added to provide redundancy and failover should other servers in the partition fail. When multiple servers participate in the same partition, the servers are referred to as "replicas" because each server maintains a duplicate copy of the partition's call state. For example, if a two-partition system has two servers in the first partition, each server manages a replica of call states A through M. If one or more servers in a partition fails or is disconnected from the network, any available replica can automatically provide call state data to the engine tier. The SIP data tier can have a maximum of three replicas, providing two levels of redundancy.

See Chapter 6, "Configuring SIP Data Tier Partitions and Replicas" for more information about configuring the SIP data tier for high availability.

Note: Because the engine tier relies on SIP data tier servers in order to retrieve call state data, Oracle recommends using dual Network Interface Cards (NICs) on engine and SIP data tier machines to provide redundant network connections.

15.4.1 Example of Writing and Retrieving Call State Data

When an initial SIP message is received, Oracle WebLogic Communication Services uses Servlet mapping rules to direct the message to the appropriate SIP Servlet deployed in the engine tier. The engine tier maintains no stateful information about SIP dialogs, but instead persists the call state to the engine tier at SIP transaction boundaries. A hashing algorithm is applied to the call state to select a single SIP data tier partition in which to store the call state data. The engine tier server then "writes" the call state to each replica within that partition and locks the call state. For example, if the SIP data tier is configured to use two servers within each partition, the engine tier opens a connection to both replicas in the partition, and writes and locks the call state on each replica.

In a default configuration, the replicas maintain the call state information only in memory (available RAM). Call state data can also be configured for longer-term storage in an RDBMS, and it may also be persisted to an off-site Oracle WebLogic Communication Services installation for geographic redundancy.

When subsequent SIP messages are generated for the SIP dialog, the engine tier must first retrieve the call state data from the SIP data tier. The hashing algorithm is again applied to determine the partition that stores the call state data. The engine tier then asks each replica in the partition to unlock and retrieve the call state data, after which a Servlet on the engine tier can update the call state data.

15.4.2 RDBMS Storage for Long-Lived Call State Data

Oracle WebLogic Communication Services enables you to store long-lived call state data in an Oracle RDBMS in order to conserve RAM. The SIP data tier persists a call

state's data to the RDBMS after the call dialog has been established, and retrieves or deletes the persisted call state data as necessary to modify or remove the call state. See Section 6.4, "Storing Long-Lived Call State Data In A RDBMS".

15.5 Geographically-Redundant Installations

Oracle WebLogic Communication Services can be installed in a geographically-redundant configuration for customers who have multiple, regional data centers, and require continuing operation even after a catastrophic site failure. The geographically-redundant configuration enables multiple Oracle WebLogic Communication Services installations (complete with engine and SIP data tier clusters) to replicate call state transactions between one another. If the a particular site's installation were to suffer a critical failure, the administrator could choose to redirect all network traffic to the secondary, replicated site to minimize lost calls. See Section 6.6, "Using Geographically-Redundant SIP Data Tiers."

15.6 Example Hardware Configurations

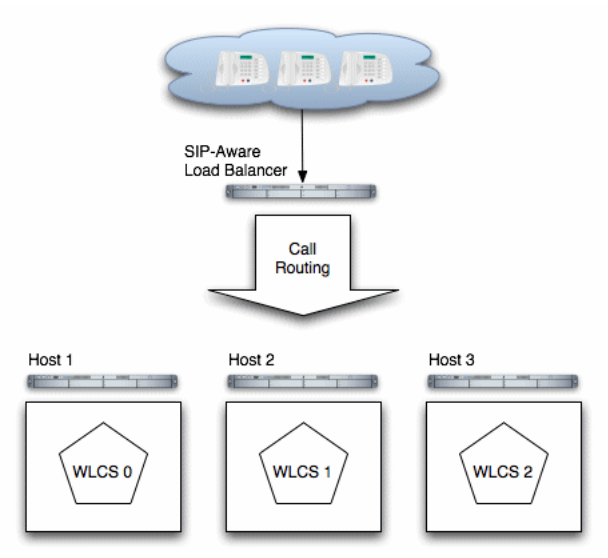
Oracle WebLogic Communication Services's flexible architecture enables you to configure engine and SIP data tier servers in a variety of ways to support high throughput and/or provide high availability.

15.7 Alternate Configurations

Not all Oracle WebLogic Communication Services requirements require the performance and reliability provided by multiple servers in the engine and SIP data tier. On a development machine, for example, it is generally more convenient to deploy and test applications on a single server, rather than a cluster of servers.

Oracle WebLogic Communication Services enables you to combine engine and SIP data tier services on a single server instance when replicating call states is unnecessary. In a combined-tier configuration, the same Oracle WebLogic Communication Services instance provides SIP Servlet container functionality and also manages the call state for applications hosted on the server. Although the combined-tier configuration is most commonly used for development and testing purposes, it may also be used in a production environment if replication is not required for call state data. Figure 15–2 shows an example deployment of multiple combined-tier servers in a production environment.

Figure 15–2 Single-Server Configurations with SIP-Aware Load Balancer



Because each server in a combined-tier server deployment manages only the call state for the applications it hosts, the load balancer must be fully "SIP aware." This means that the load balancer actively routes multiple requests for the same call to the same Oracle WebLogic Communication Services instance. If requests in the same call are not pinned to the same server, the call state cannot be retrieved. Also keep in mind that if a Oracle WebLogic Communication Services instance fails in the configuration shown in Figure 15–2, all calls handled by that server are lost.

16

Deployment Topologies for Communication Services

This chapter describes the OWLCS deployment topologies in the following sections:

16.1 Terminology

The following terms are used in this chapter:

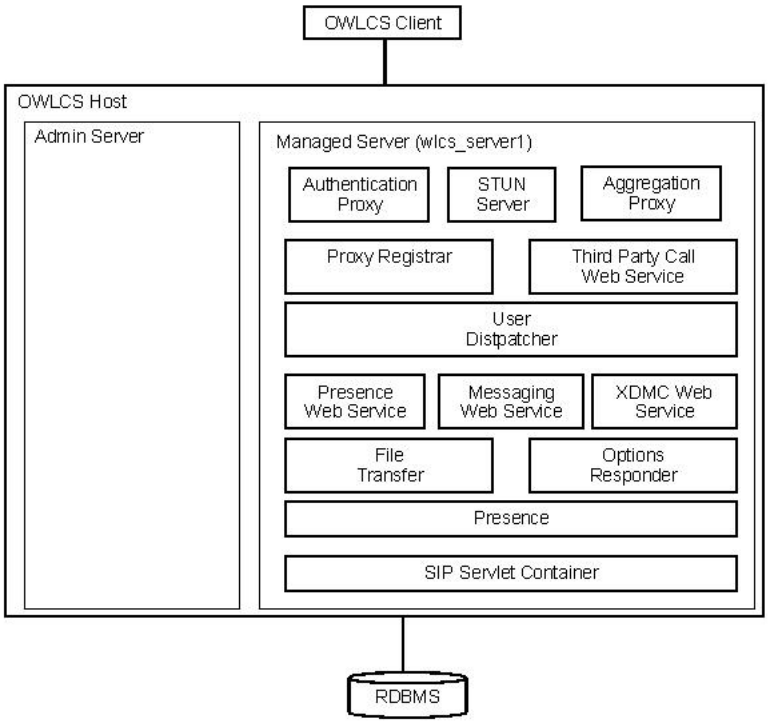
- Administration Server--a special server instance within a domain to configure and manage all resources in the domain. There is just one Administration Server in a domain.
- Domain--a logically related group of server resources
- High Availability (HA)--refers to a system or component that is continuously operational for a desirable long length of time
- JVM--Java Virtual Machine
- Managed Server--a server instance within a domain where resources are deployed. There can be one or more managed servers within a domain.
- Node--a physical machine
- OWLCS Client--a client application that can connect to OWLCS server and use it's features -Presence, Instant Messaging (IM), VoIP, Presence Web Service, and Messaging Web Service. For example: Oracle Communicator.

16.2 OWLCS Deployment Topologies

OWLCS supports single-node and multi-node deployment topologies:

- All-In-One Managed Server--All OWLCS components are deployed on a Managed Server within a domain on a physical machine. An Administration Server installed on the same domain is used to configure and manage OWLCS components.

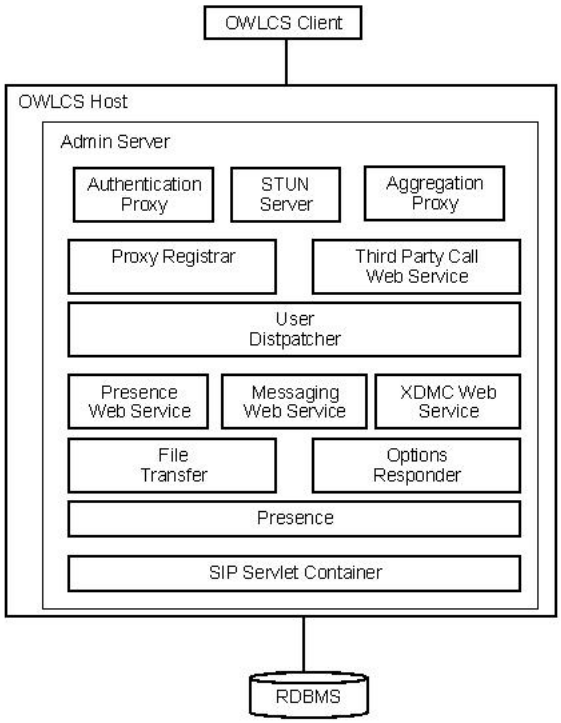
Figure 16–1 All-in-One Managed Server



See Section 16.2.1, "Single-node Topologies" for more information.

- All-In-One Administration Server--All OWLCS components are deployed on an Administration Server within a domain on a physical machine. There are no managed servers. Configuration and management of OWLCS components is done using the same Administration Server.

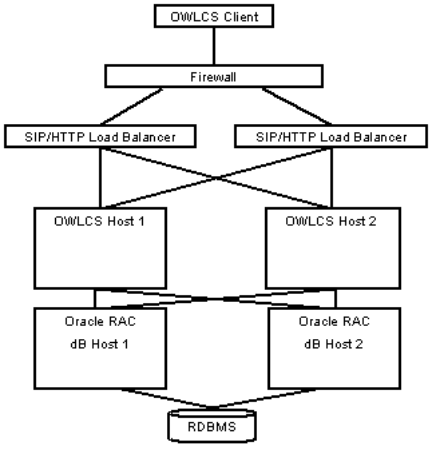
Figure 16–2 All-in-One Administration Server



See Section 16.2.1, "Single-node Topologies" for more information.

- Enterprise Deployment--This is the minimal recommended HA topology for Enterprise Deployment for OWLCS. OWLCS components are separated as SIP Infrastructure, Services and Presence components. Each group of components is deployed within its own domain on two different machines.

Figure 16–3 Enterprise Deployment



See Section 16.3.1, "Introduction to OWLCS Enterprise Deployment Topology" for more information.

16.2.1 Single-node Topologies

Both the All-In-One Administration Server and All-In-One Managed Server Topologies are intended to provide an out-of-box experience in which Presence, Instant Messaging, Voice-Over-IP, Third Party Calling, and Presence and MessagingWeb Services work with an OWLCS client. These topologies are recommended for use in a test and evaluation environment, not in a production environment. They do not include High-Availability support.

In an All-In-One Administration Server topology, the one and the only server (the Administration Server) runs in a single JVM. In an All-In-One Managed Server topology, the Managed Server runs in one JVM and the Administration Server runs in a separate JVM.

16.3 Oracle WebLogic Communication Services Enterprise Deployment Topology

This section describes the Oracle WebLogic Communication Services (OWLCS) high-availability topology: Enterprise Deployment. The following sections are included:

- Section 16.3.1, "Introduction to OWLCS Enterprise Deployment Topology"
- Section 16.3.2, "Geographic Redundancy"
- Section 16.3.3, "Failover"

16.3.1 Introduction to OWLCS Enterprise Deployment Topology

OWLCS supports a multi-node deployment topology. A *SIP Container cluster* is defined as a set of SIP Container instances that share state related to the applications. A cluster consists of multiple nodes, with each node running one instance of OWLCS.

A highly available OWLCS cluster provides the following:

- Replication of objects and values contained in a SIP Application Session
- Database backed location service data
- Load balancing of incoming requests across OWLCS SIP nodes
- Overload protection protects the server from malfunctioning in the event of overload and rejects traffic which cannot be handled properly.
- Transparent failover across applications within the cluster. If an instance of an application fails, it becomes unresponsive and the session can fail over to another instance of the application, on another node in a cluster.

Note: For more information on OWLCS as a scalable Presence deployment, see Appendix G, "Deploying a Scalable Presence Deployment"

Note: For information on installing and configuring OWLCS Enterprise Deployment topology, see *Oracle WebLogic Communication Services Installation Guide*.

16.3.1.1 Runtime Processes

Oracle WebLogic Communication Services start scripts use default values for many JVM parameters that affect performance. For example, JVM garbage collection and heap size parameters may be omitted, or may use values that are acceptable only for evaluation or development purposes. In a production system, you must rigorously profile your applications with different heap size and garbage collection settings in order to realize adequate performance. See *Oracle WebLogic Communication Services Administrator's Guide* for suggestions about maximizing JVM performance in a production domain.

Because a typical Oracle WebLogic Communication Services domain contains numerous engine and servers, with dependencies between the different server types, you should generally follow this sequence when starting up a domain:

1. **Start the Administration Server for the domain.** Start the Administration Server in order to provide the initial configuration. The Administration Server can also be used to monitor the startup/shutdown status of each Managed Server. You generally start the Administration Server by using either the `startWebLogic.sh/cmd` script installed with the Configuration Wizard, or a custom startup script.
2. **Start Managed Servers.** Next you can start managed servers using the `startManagedWebLogic.sh/cmd` script or a custom startup script.

16.3.1.2 Request Flow

Request Flow is described in *JSR 289*. This specification is an enhancement to the SIPServlet specification.

For details on *JSR 289*, see:
http://www.oracle.com/technology/tech/java/standards/jsr289/index.html

16.3.1.3 Client Connections

The default HTTP network configuration for each Oracle WebLogic Communication Services instance is determined from the *Listen Address* and *Listen Port* setting for each server. However, Oracle WebLogic Communication Services does not support the SIP protocol over HTTP. The SIP protocol is supported over the UDP and TCP transport protocols. SIPS is also supported using the TLS transport protocol.

To enable UDP, TCP, or TLS transports, you configure one or more *network channels* for an Oracle WebLogic Communication Services instance. A network channel is a configurable WebLogic Server resource that defines the attributes of a specific network connection to the server instance. Basic channel attributes include:

- Protocols supported by the connection,
- Listen address (DNS name or IP address) of the connection,
- Port number used by the connection,
- (optional) Port number used by outgoing UDP packets,
- Public listen address (load balancer address) to embed in SIP headers when the channel is used for an outbound connection.

You can assign multiple channels to a single Oracle WebLogic Communication Services instance to support multiple protocols or to utilize multiple interfaces available with multi-homed server hardware. You cannot assign the same channel to multiple server instances.

When you configure a new network channel for the SIP protocol, both the UDP and TCP transport protocols are enabled on the specified port. You cannot create a SIP channel that supports only UDP transport or only TCP transport. When you configure a network channel for the SIPS protocol, the server uses the TLS transport protocol for the connection.

As you configure a new SIP Server domain, you will generally create multiple SIP channels for communication to each engine tier server in your system. Engine tier servers can communicate to SIP state tier replicas using the configured Listen Address attributes for the replicas. Note, however, that replicas must use unique Listen Addresses in order to communicate with one another.

16.3.1.4 Artifacts

Installation, configuration and deployment create the following artifacts for OWLCS.

16.3.1.4.1 .ear Files .ear files for deploying your applications are found in your middleware home directory (for example): `$MW_HOME/as11gr1wlcs1/communications/applications`. They are typically easily-identified by their names (for example: `sdpmessagingdriver-smpp.ear` for deploying SMPP driver).

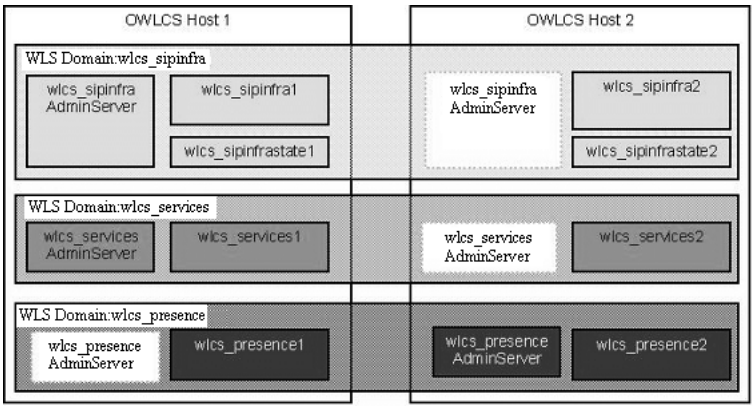
16.3.1.4.2 Configuration Files Configuring components is accomplished through various .xml files. They can be found in your middleware home directory (for example): `$DOMAIN_HOME/config/communications`. They are typically easily-identified by their names (for example: `usermessagingdriver-smpp_Plan.xml` for configuring SMPP).

16.3.1.4.3 Log Files Log activities are stored in log files. There are many log files, but of special note are install, activity, error, and diagnostic logs. They are found in your middleware home directory (for example): `$DOMAIN_HOME/servers/wlcs_server1/logs`.

16.3.1.5 Topology Components

Components of a highly available OWLCS topology are detailed below. Figure 16–4 shows details of the OWLCS hosts that support the topology.

Figure 16–4 Host details for OWLCS

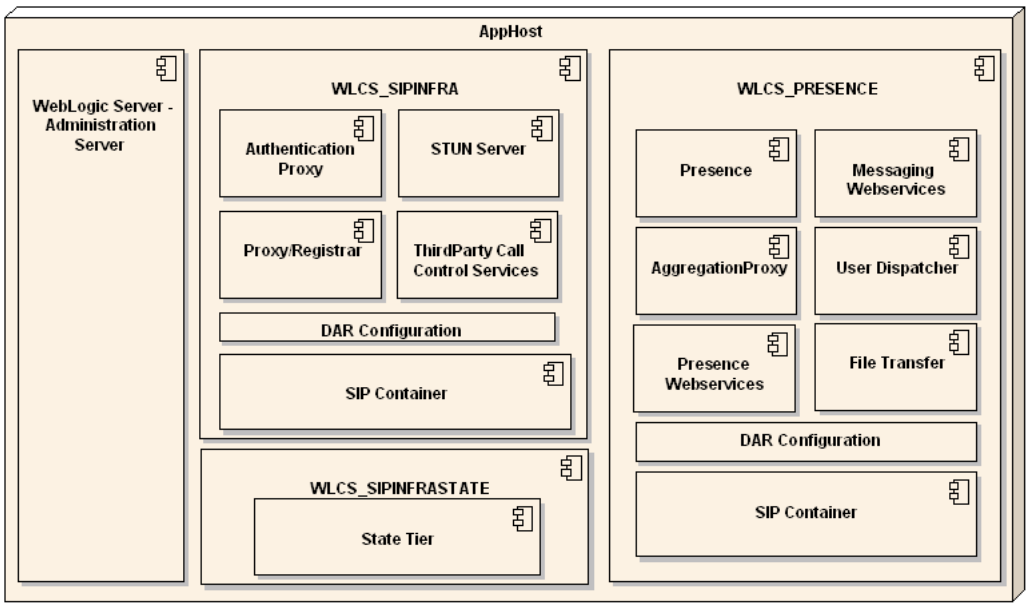


- Section 16.3.1.5.1, "SIP Containers"
- Section 16.3.1.5.2, "Third-Party Load Balancer"

- Section 16.3.1.5.3, "Proxy Registrar"
- Section 16.3.1.5.4, "Presence Services"
- Section 16.3.1.5.5, "Presence Web Services"
- Section 16.3.1.5.6, "Third-Party Call Control Web Services"
- Section 16.3.1.5.7, "Aggregation Proxy"
- Section 16.3.1.5.8, "Authentication Proxy"
- Section 16.3.1.5.9, "File Transfer Service"
- Section 16.3.1.5.10, "Messaging Services"
- Section 16.3.1.5.11, "STUN Service"
- Section 16.3.1.5.12, "DAR Configuration"
- Section 16.3.1.5.13, "Oracle Communicator Client"
- Section 16.3.1.5.14, "State Tier"
- Section 16.3.1.5.15, "User Dispatcher"
- Section 16.3.1.5.16, "Oracle RAC Database"

Note: For more information on configuring these components, see *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter*.

Figure 16–5 OWLCS High Availability detail



16.3.1.5.1 SIP Containers OWLCS extends the core WebLogic Server platform with a SIP Container compliant with JSR 289. This enables the development of J2EE applications that processes SIP in addition to HTTP for any advanced communications application. The platform enables the development of complementary communications services that integrate with SIP-based IP-PBXs as well as other SIP

elements such as standard SIP clients. For more information on SIP Container, see *Oracle WebLogic Communication Services Administrator's Guide*.

Two or more SIP Containers are linked to one another through OWLCS SIP state replication. The OWLCS SIP state on each computer is replicated to other nodes in the SIP State Tier so that if one SIP Container node fails, another container node takes over, using the replicated state of the failed node.

16.3.1.5.2 Third-Party Load Balancer A third-party load balancer balances the load of incoming traffic. It also redirects traffic from failed nodes to operational ones. Your Load Balancer must be capable of routing both HTTP and SIP traffic, and must be configured to do so. For more information about Load Balancers, see *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle WebCenter*.

16.3.1.5.3 Proxy Registrar The OWLCS Proxy Registrar combines the functionality of a SIP Proxy Server and Registrar. Its main tasks include registering subscribers, looking up subscriber locations, and proxying requests onward. The Proxy Registrar stores user location and registration data on the Oracle database. This is an optional component. For more information on Proxy Registrar, see Section 14.1, "Proxy Registrar".

16.3.1.5.4 Presence Services OWLCS includes a Presence Service that acts as the aggregator of presence information and provides a subscribe/notify for applications and end-users to consume presence information. An application can integrate either using web services or by using a compliant SIP-based end-user client.

16.3.1.5.5 Presence Web Services OWLCS enables Web Service clients to access presence services through its support of the Parlay X Presence Web Service as defined in *Open Service Access, Parlay X Web Services, Part 14, Presence ETSI ES 202 391-14*. A Parlay X Web Service enables an HTTP Web Service client to access such presence services as publishing and subscribing to presence information. The Parlay X Presence Web Service does not require developers to be familiar with the SIP protocol to build such a Web-based client; instead, Parlay X enables Web developers to build this client using their knowledge of Web Services.

16.3.1.5.6 Third-Party Call Control Web Services The Third Party Call Parlay X 2.1 communication services implement the Parlay X 2.1 Third Party Call interface, (Standards reference: ETSI ES 202 391-2 V1.2.1 (2006-12), Open Service Access (OSA); Parlay X Web Services; Part 2: Third Party Call (Parlay X 2). For more information on Third-Party Call Control Web Services, see *Oracle WebLogic Communication Services Developer's Guide*.

16.3.1.5.7 Aggregation Proxy The Aggregation Proxy authorizes web service calls and authenticates XCAP traffic. The Aggregation Proxy then proxies this traffic to the Parlay X Web Service and XDMS. This is an optional component.

16.3.1.5.8 Authentication Proxy The Authentication Proxy is a SIP application that upon successful authentication populates a *P-Asserted-Identity* header into the request. The authentication itself is not performed by the Authentication Proxy application. SIP digest-based authentication is configured in a standard way, as for all SIP applications. The *P-Asserted-Identity* header value will be the externally stored (for example, in Oracle Identity Management or Database) preferred SIP- and/or TEL-URL for the authenticated user.

16.3.1.5.9 File Transfer Service OWLCS includes an Oracle-proprietary file transfer service enabling users to transfer files to one-another.

16.3.1.5.10 Messaging Services Messaging Services implements support for a subset of the operations in the *SendMessage*, *ReceiveMessage*, and MessageNotificationManager interfaces, as they are defined in *ETSI ES 202 391-5 V1.2.1 (2006-12), Open Service Access (OSA), Parlay X Web Services Part 5: Multimedia Messaging (Parlay X 2)*.

16.3.1.5.11 STUN Service The OWLCS STUN Service implements STUN (Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)). For more information on STUN Service, see Section 14.2, "STUN Service".

16.3.1.5.12 DAR Configuration DAR Configuration. Application Router is a SIP application that routes incoming SIP requests to the correct application. The Application Router routes requests by placing route headers in each SIP request it processes. A number of route headers can be placed in a request, each representing a different destination URI. The SIP request is either sent through the chain of destination URIs, or proxied to a new URI upon arriving at its first destination.

16.3.1.5.13 Oracle Communicator Client Oracle Communicator is a client communication application that enables users to keep in touch with others in their organization. You can see your Contacts' presence (that is, their availability), and communicate with them by sending instant messages and sharing files.

16.3.1.5.14 State Tier The Oracle WebLogic Communication Services SIP state tier node manages the application call state for concurrent SIP calls. The SIP state tier may manage a single copy of the call state or multiple copies across the cluster as needed to ensure that call state data is not lost if a server machine fails or network connections are interrupted.

16.3.1.5.15 User Dispatcher The User Dispatcher enables the Presence and XDMS applications to scale. The User Dispatcher is a proxy that dispatches SIP and XCAP (over HTTP) requests to their appropriate destinations on a consistent basis.

Because the Presence application maintains the states for all users in a deployment, User Dispatcher enables scaling (distribution) of the Presence application. The User Dispatcher supports request dispatching to the Presence Server and XDMS sub-applications, which use the SIP and XCAP (over HTTP) protocols.

16.3.1.5.16 Oracle RAC Database Oracle RDBMS should be installed and operational in an Oracle RAC environment. The supported versions are 10.2.0.4 and 11.1.0.7. Refer to *Oracle Clusterware Installation Guide 11g Release 1 (11.1.)*, *Oracle Real Application Clusters: For 11g Release 1 (11.1)* and *Oracle Real Application Clusters Installation Guide 11g Release 1*.

16.3.1.6 Overview of SIP State Tier Configuration

The Oracle WebLogic Communication Services SIP state tier is a cluster of server instances that manages the application call state for concurrent SIP calls. The SIP state tier may manage a single copy of the call state or multiple copies as needed to ensure that call state data is not lost if a server machine fails or network connections are interrupted.

The SIP state tier cluster is arranged into one or more *partitions*. A partition consists of one or more SIP state tier server instances that manage the same portion of concurrent call state data. In a single-server Oracle WebLogic Communication Services installation, or in a two-server installation where one server resides in the engine tier and one resides in the SIP state tier, all call state data is maintained in a single partition. Multiple partitions are required when the size of the concurrent call state exceeds the maximum size that can be managed by a single server instance. When

more than one partition is used, the concurrent call state is split among the partitions, and each partition manages an separate portion of the data. For example, with a two-partition SIP state tier, one partition manages the call state for half of the concurrent calls (for example, calls A through M) while the second partition manages the remaining calls (N through Z).

In most cases, the maximum call state size that can be managed by an individual server is limited by the heap size in the Java Virtual Machine.

Additional servers can be added within the same partition to manage copies of the call state data. When multiple servers are members of the same partition, each server manages a copy of the same portion of the call data, referred to as a *replica* of the call state. If a server in a partition fails or cannot be contacted due to a network failure, another replica in the partition supplies the call state data to the engine tier. Oracle recommends configuring two servers in each partition for production installations, to guard against machine or network failures. A partition can have a maximum of three replicas for providing additional redundancy.

16.3.1.7 Example SIP State Tier Configurations and Configuration Files

The sections that follow describe some common Oracle WebLogic Communication Services installations that utilize a separate SIP state tier.

16.3.1.7.1 SIP State Tier with One Partition A single-partition, two server SIP state tier represents the simplest state tier configuration.

For example, the `datatier.xml` configuration file shown in Example 16–1 creates a two-replica configuration.

Example 16–1 SIP State Tier Configuration for Small Deployment with Replication

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http:....">
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
      <server-name>DataNode0-1</server-name>
    </partition>
  </data-tier>
```

16.3.1.7.2 SIP State Tier with Two Partitions Multiple partitions can be created by defining multiple `partition` entries in `datatier.xml`, as shown in Example 16–2.

Example 16–2 Two-Partition SIP State Tier Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http:....">
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
    </partition>
    <partition>
      <name>Partition1</name>
      <server-name>DataNode1-0</server-name>
    </partition>
  </data-tier>
```

16.3.1.7.3 SIP State Tier with Two Partitions and Two Replicas Replicas of the call state can be added by defining multiple SIP state tier servers in each partition. Example 16–3

shows the `datatier.xml` configuration file used to define a system having two partitions with two servers (replicas) in each partition.

Example 16–3 SIP State Tier Configuration for Small Deployment

```
<?xml version="1.0" encoding="UTF-8"?>
  <data-tier xmlns="http:....">
    <partition>
      <name>Partition0</name>
      <server-name>DataNode0-0</server-name>
      <server-name>DataNode0-1</server-name>
    </partition>
    <partition>
      <name>Partition1</name>
      <server-name>DataNode1-0</server-name>
      <server-name>DataNode1-1</server-name>
    </partition>
  </data-tier>
```

16.3.1.8 Storing Long-Lived Call State Data in an RDBMS

Oracle WebLogic Communication Services enables you to store long-lived call state data in an Oracle RDBMS in order to conserve RAM. When you enable RDBMS persistence, by default the SIP state tier persists a call state's data to the RDBMS after the call dialog has been established, and at subsequent dialog boundaries, retrieving or deleting the persisted call state data as necessary to modify or remove the call state.

Oracle also provides an API for application designers to provide *hints* as to when the SIP state tier should persist call state data. These hints can be used to persist call state data to the RDBMS more frequently, or to disable persistence for certain calls.

Oracle WebLogic Communication Services can use the RDBMS to supplement the SIP state tier's in-memory replication functionality. To improve latency performance when using an RDBMS, the SIP state tier maintains SIP timers in memory, along with call states being actively modified (for example, in response to a new call being set up). Call states are automatically persisted only after a dialog has been established and a call is in progress, at subsequent dialog boundaries, or in response to persistence hints added by the application developer.

When used in conjunction with an RDBMS, the SIP state tier (the term *state tier* is used interchangeably with *data tier* in this document) selects one replica server instance to process all call state writes (or deletes) to the database. Any available replica can be used to retrieve call states from the persistent store as necessary for subsequent reads.

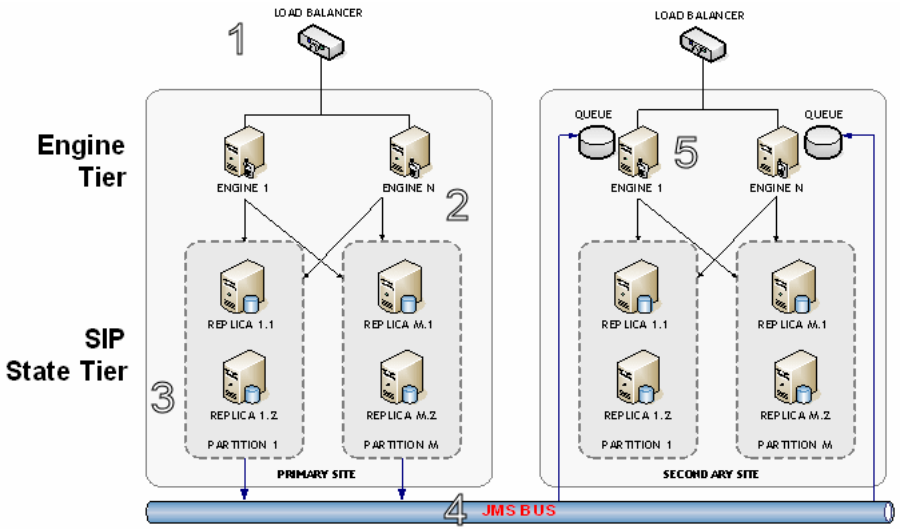
RDBMS call state storage can be used in combination with an engine tier cache, if your domain uses a SIP-aware load balancer to manage connections to the engine tier.

16.3.2 Geographic Redundancy

Geographic Redundancy ensures uninterrupted transactions and communications for providers, using geographically-separated SIP server deployments.

A primary site can process various SIP transactions and communications and upon determining a transaction boundary, replicate the state data associated with the transaction being processed, to a secondary site. Upon failure of the primary site, calls are routed from the failed primary site to a secondary site for processing. Similarly, upon recovery, the calls are re-routed back to the primary site. See *Oracle WebLogic Communication Services Administrator's Guide* for more information

Figure 16–6 Geo-Redundancy



In the preceding figure, Geo-Redundancy is portrayed. The process proceeds in this manner:

1. Call is initiated on a primary Cluster site, call setup and processing occurs normally.
2. Call is replicated as usual to the site's SIP State Tier, and becomes eligible for replication to a secondary site.
3. A single replica in the SIP State Tier then places the call state data to be replicated on a JMS queue configured on a replica site.
4. Call is transmitted to one of the available engines using JMS over WAN.
5. Engines at the secondary site monitor their local queue for new messages. Upon receiving a message, an Engine in the secondary site persists the call state data and assigns it the site ID value of the primary site.

16.3.3 Failover

Oracle's High Availability solutions include failover support for all levels of your system. During failover, the functions of a component that is malfunctioning or non-operational are addressed by a standby or replacement component. This is done without manual intervention, and in most cases end users will not be able to detect that the system is taking failover actions. There are two main types of failover: *Session Failover* and *Service Failover*.

Session Failover occurs when the session, connection, or power are interrupted. During Session Failover, ongoing calls or requests are handled by backup nodes, and users do not detect that the condition arose. The SIP Container provides services to upper-levels to help recover from session failure. The SIP protocol rebuilds state at certain pre-defined times (for example, every hour). This is designed to protect unreliable networks.

When *Service Failover* occurs (such as when a node in a cluster of nodes goes down), the load is handled by other nodes in the cluster. If an individual request is being processed at the time, it will fail, but subsequent requests will be picked up by the functioning nodes.

In its High Availability configuration, services running in the *wlcs-services* domain such as Proxy Registrar, Third-Party Call Control, and other distributable applications support Session Failover, while services running in the *wlcs_presence* domain such as Presence and other non-distributable applications support Service Failover.

16.3.3.1 OWLCS Presence Failover

OWLCS Presence, when multiple nodes are used in a High Availability deployment, distributes its services across nodes (servers). When a request for the Presence information for an entity (E1) is made, the request goes to node 1 (N1). If N1 goes down during the request, the user will see a failure. Upon resubmitting the request, it (the request) will be handled by node 2 (N2). No failure occurs during the re-request, or during any other requests that are made after the initial failure.

Fail-over is a technique that can be used by the User Dispatcher to assert a higher level of availability of the Presence Server. Since the Presence server does not replicate any state (such as established subscriptions) the state has to be recreated by the clients on the new server node by setting up new subscriptions. Also, since a subscription is a SIP dialog and the User Dispatcher is not record routing, it cannot fail-over a subscription from one node to another. All subsequent requests will follow the route set and end up on the "old" node.

This is not a problem when failing over from a "failing" server since that node is not processing the traffic anyway and any request within a dialog will eventually get a fail response or timeout and the dialog will be terminated. However, when migrating back a user from the backup node to the original node (when it has been repaired), which has to be done to maintain an even distribution after the failure, this is a problem that can lead to broken presence functionality. The only way to migrate a subscription from one running server to another is to either restart the client or the server.

However, the server that holds the subscription can actively terminate it by sending out a terminating NOTIFY and discarding the subscription state. This will force the client to issue a new initial SUBSCRIBE to establish a new dialog. For a subscription to migrate from one live node to another the User Dispatcher must fail-over the traffic (which is only affecting initial requests) and instruct the current server to terminate the subscriptions.

16.3.3.2 Presentity Migration

Presentities must be migrated when the set of nodes have changed. This involves having the Presence application to terminate some or all subscriptions to make the migration happen.

16.3.3.2.1 Stateless User Dispatcher and Even Distribution The most basic approach is to contact the Presence application on all nodes to terminate all its subscriptions. The problem with this is that a burst of traffic will be generated although spread out over a period of time. This time period results in incorrect presence states since the longer the termination period is the longer it will take until all users get a correct presence state.

To optimize this you could terminate only those subscriptions that actually need to be terminated (the ones that has been migrated). The problem is that the User Dispatcher does not know which users these are (since it does stateless distribution based on an algorithm) and the Presence application does not either (since it only knows what users it has). However, if the Presence application could iterate over all its subscriptions and for each of them ask the User Dispatcher if this user would go to this Presence node, then the Presence server could terminate only those that will not come back to itself. This may be a heavy operation, but under the constraint that each

Presence server is collocated with a User Dispatcher each such callback would be within the same JVM.

16.3.3.2.2 Presence Application Broadcast Another solution is to have the Presence servers guarantee that a user only exists on one Presence node at any given time. This can be done by having the Presence application broadcast a message to all its neighbors when it receives a PUBLISH or SUBSCRIBE for a new presentity (a presentity that it does not already have a state for). If any other Presence node that receives this broadcast message already has active subscriptions for this presentity, that server must terminate that subscription so that the client can establish a new subscription with the new server.

With this functionality in the Presence application, the User Dispatcher would not have to perform additional steps to migrate a user from one live node to another.

16.3.3.3 Standby Server Pool

Another approach is to have a standby pool of servers that are idling ready to take over traffic from a failing node. When an active node fails the User Dispatcher will redistribute all its traffic to one server from the standby pool. This node will now become active and when the failing node eventually is repaired it will be added to the standby pool. This will eliminate the need for migrating users “back” from a live node when a failing node resumes.

This approach requires more hardware and the utilization of hardware resources will not be optimal.

16.3.3.4 Failure Types

There are several types of failures that can occur in a Presence server and different types of failures may require different actions from the User Dispatcher.

16.3.3.4.1 Fatal Failures If the failure is fatal all state information is lost and established sessions will fail. However, depending on the failure response, subscriptions (presence subscribe sessions) can survive using a new SIP dialog. If the response code is a 481 the presence client must according to RFC 3265 establish a new SUBSCRIBE dialog and this is not considered to be a failure from a presence perspective. All other failure responses may (depending on the client implementation) be handled as an error by the client and should therefore be considered a failure.

After a fatal failure the server does not have any dialog states from the time before the failure, which means that all subsequent requests that arrive at this point will receive a 481 response back. During the failure period all transactions (both initial and subsequent) will be terminated with a non-481 error code, most likely a 500 or an internal 503 or 408 (depending on if there is a proxy in the route path or not, and what the nature of the failure is).

Typically a fatal failure will result in the server process or the entire machine being restarted.

16.3.3.4.2 Temporary Failures A temporary failure is one where none or little data is lost so that after the failure session states will remain in the server. This means that a subsequent request that arrives after the server has recovered from the failure will be processed with the same result, as it would have been before the failure.

All requests that arrive during the failure period will be responded with a non-481 failure response, such as 503.

In general a temporary failure has a shorter duration, and a typical example is an overload situation in which case the server will respond 503 on some or all requests.

16.3.3.5 Failover Actions

The User Dispatcher can take several actions when it has detected a failure in a Presence server node. The goal with the action is to minimize the impact of the failure in terms of number of failed subscriptions and publications and the time it takes to recover. In addition to this the User Dispatcher needs to keep the distribution as even as possible over the active servers.

The fail-over action to be used in this version of the User Dispatcher is to disable the node in the pool. This approach is better than removing the node because when the ResizableBucketServerPool is used since the add and remove operations are not deterministic. This means that the result of adding a node depends on the sequence of earlier add and delete operations, whether as the disable operation will always result in the same change in distribution given the set of active and disabled nodes.

16.3.3.6 Overload Policy

An activated overload policy can indicate several types of failures but its main purpose is to protect from a traffic load that is to big for the system to handle. If such a situation is detected as a failure, fail-over actions can lead to bringing down the whole cluster since if the distribution of traffic is fairly even all the nodes will be in or near an overloaded situation. If the dispatchers remove one node from the cluster and redistribute that node's traffic over the remaining nodes they will certainly enter an overload situation that causes a chain reaction.

Since it is difficult to distinguish this overload situation from a software failure that triggers the overload policy to be activated even though the system is not under load, it might still be better to take the fail-over action unless Overload Policy is disabled. If the system is really in an overload situation it is probably under dimensioned and then the fail-over should be disabled.

The User Dispatcher will not fail over when it has detected a 503 response (which indicates overload policy activated). However, if a server is in the highest overload policy state where it drops messages instead of responding 503 the User Dispatcher monitor will receive an internal 408, which can never be distinguished from a dead server and failover will occur.

16.3.3.7 Synchronization of Failover Events

Depending on the failure detection mechanism there may be a need to synchronize the fail-over events (or the resulting state) between the different dispatcher instances. This is required if the detection mechanism is not guaranteed to be consistent across the cluster, such as an Error Response. For instance one server node sends a 503 response on one request but after that works just fine (this can be due to a glitch in the overload policy). If there was only one 503 sent then only one dispatcher instance will receive it and if that event triggers a fail-over then that dispatcher instance will be out of sync with the rest of the cluster. Further, even if the grace period is implemented so that it takes several 503 responses over a time period to trigger the fail-over there is still a risk for a race condition if the failure duration is the same as the grace period.

The following methods can be used to assure that the state after fail-over is synchronized across the cluster of dispatcher instances:

16.3.3.7.1 Broadcasting Fail-Over Events In this approach each dispatcher instance have to send a notification to all other instances (typically using JGroups or some other multicast technique) when it has decided to take a fail-over action and change the set

of servers. This method can still lead to race conditions since two instances may fail-over and send a notification at the same time for two different server nodes.

16.3.3.7.2 Shared State If all dispatcher nodes in the cluster share the same state from a "single source of truth" then when the state is changed (due to a fail-over action) by any instance all other instances will se the change.

16.3.3.8 Expanding the Cluster

Since the Presence application can generate an exponentially increasing load due to the fact that every user subscribes to multiple (potentially a growing number of) other users, there is a need for a way to dynamically expand the cluster without too much disturbance. Compared to for instance a classic telecom application where it may be acceptable to bring all servers down for an upgrade of the cluster during low traffic hours, a Presence system may have higher availability requirements than that.

Expanding the cluster may involve both adding Presence nodes and User Dispatcher nodes.

When a new Presence server is added to a cluster, some presentites must be migrated from old nodes to the new node in order to keep a fairly even distribution. This migration needs to be minimized to avoid a too big flood of traffic on the system upon changing the cluster.

When a new User Dispatcher is added to the cluster that User Dispatcher node must achieve the same dispatching state as the other dispatcher nodes. This may depending on the pool implementation require a state being synchronized with the other dispatcher nodes (for instance when using the bucket pool implementation with persistence).

Note: Each User Dispatcher within the Presence Cluster must be configured to include all the Presence Server instances in the cluster in its list of presence servers to which they will dispatch.

16.3.3.8.1 Updating the Node Set Depending on the algorithm used to find the server node for a given presentity, different number of presentity will be "migrated" to another node when a new node is added or removed. An optimal Pool implementation will minimize this number.

16.3.3.8.2 Migrating Presentities When the node set has been updated some Presentites may have to be migrated to maintain an even distribution. The different ways to do this are described in "Presentity Migration".

16.3.3.9 Failover Use Cases

These use cases illustrates how the User Dispatcher reacts in different failure situations in one or several Presence server nodes.

16.3.3.9.1 One Presence Server Overloaded for 60 Seconds The cluster consists of four Presence servers, each node consisting of one OWLCS instance with a User Dispatcher and a Presence application deployed. 100,000 users are distributed over the four servers evenly (25,000 on each node). Due to an abnormally long GC pause on one of the servers, the processing of messages is blocked by the Garbage Collector, which leads to the SIP queues getting filled up and the overload policy is activated. 60s later the processing resumes and the server continues to process messages.

The User Dispatcher will not do any fail-over but keep sending traffic to the failing node. In this case no sessions will be migrated to another node since all PUBLISH and initial SUBSCRIBE requests will be sent to the failing node. The initial SUBSCRIBES that arrives during the failure period will fail with a non-481 error (likely 503). It is up to the client to try and setup a new subscription when the failing one expires or report a failure. All PUBLISH requests and initial SUBSCRIBE request will generate a failure.

When the failing node resumes to normal operation all traffic will be processed again and no requests should fail. The time it takes until all presence states are “correct” again will be minimal since no sessions were failed-over.

If the monitoring feature is implemented in a way that detects the node as “down” in this case, then some users will be migrated to another node and when this node comes back they will be migrated back again. This will generate some increased load for a duration of time. If the overload policy was activated because of a too high traffic load this migration is bad, since is will most likely happen again and since the other servers will most likely also be close to overload. This could lead to a chain reaction resulting in the whole cluster going down and a complete loss of service.

16.3.3.9.2 One Presence Server Overloaded Multiple Times for Five Seconds This use case describes a Presence server that is going in and out from overload with short time periods such as 5 seconds. This is common if the system is under dimensioned and can barely cope with the traffic load, but it could also be caused by some other disturbance only on that particular node. The User Dispatcher will behave exactly as in "One Presence Server Overloaded for 60 Seconds" and the result will be the same except that the number of failed sessions and failed-over sessions will be smaller due to the shorter failure period.

16.3.3.9.3 Overload Policy Triggered by an OWLCS Software Failure A failure in the OWLCS software or an application deployed on top of it causes all threads to be locked (deadlock). This will eventually lead to that the in queue is filled up and the overload policy is activated even though the system is not actually overloaded. This is a permanent error that can only be solved by restarting the server.

Depending on if and how the monitor function is implemented the number of affected users can be minimized. However this cannot be distinguished from a “real” overload situation in which case a fail-over may not be the best thing to do.

16.3.3.9.4 A Presence Server Hardware Failure The cluster consists of four Presence servers, each node consisting of one OWLCS instance with a User Dispatcher and a Presence application deployed. 100,000 users are distributed over the four servers evenly (25,000 on each node). One of the presence servers crashes due to a hardware failure. A manual operation is required to replace broken server with a new one and only after two hours is the server up and running again. Depending on the type of the failure the response code sent back on transactions proxied to the failed node will be 408 or 503.

In this case all sessions on this node will fail since the failure duration is (most likely) more than the expiration time for the subscriptions. If a monitor server is implemented with fail-over then the failure time will be minimized to the detection time (seconds). The users will be migrated by the migration feature, which will create an increased load for a duration of time.

Because the User Dispatcher was also running on the failed node, all the persisted data for the user dispatcher will be lost when replacing the server with a new machine.

16.3.3.9.5 Expanding the Cluster with One Presence Node The cluster consists of 3 Presence servers, each node consisting of one OWLCS instance with a User Dispatcher and a

Presence application deployed. 100,000 users are distributed over the four servers evenly (33,000 on each node). A new node is installed and added to the cluster. The following sequence of operations are performed to add the new node:

1. The User Dispatcher and the Presence application on the new node are configured with the same settings as the rest of the cluster. This includes synchronizing the distribution state to the new User Dispatcher in case of a pool implementation with persistence.
2. The addServer JMX operation is invoked with the new node on the cluster User Dispatcher MBean. This will invoke the addServer operation on all User Dispatcher nodes (including the new node).
3. The Load Balancer is reconfigured with the new node so that initial requests are sent to the new User Dispatcher node.
4. Depending on the migration approach an additional JMX operation may be invoked on the Presence application (using the cluster MBean server).

The result of this is that the new distribution of users is 25,000 on each node after 8,000 users have been migrated. Depending on the migration method this will generate an increased load of traffic on the system over a period of time.

16.3.3.9.6 Removing a Node from the Cluster The cluster consists of four Presence servers, each node consisting of one OWLCS instance with a User Dispatcher and a Presence application deployed. 100,000 users are distributed over the four servers evenly (25,000 on each node). One Presence node is removed from the cluster. The following sequence of operations are performed to remove the node:

1. The Load Balance is reconfigured to not include the node to be removed.
2. The removeNode JMX operation is invoked to remove the node from all the User Dispatcher's in the cluster. The cluster MBean is used to delegate the operation.
3. Depending on the migration approach an additional JMX operation may be invoked on the node to be removed.
4. When all users have been migrated from the node to be removed (the duration of this depends on the migration method) the node is finally stopped and removed from the cluster.

The result of this is that the new distribution of users is 33,000 on each node after 8,000 have been migrated.

16.3.3.9.7 OPMN Restart After a Presence Server Crash Consider a four-node cluster with a User Dispatcher and a Presence application deployed on each node. The Presence server JVM on one of the nodes crashes and OPMN restarts the process. The restart takes one minute.

16.3.3.9.8 503 Responses from an Application Due to a software bug or misbehavior in the application, 503 responses are sent for all incoming traffic. The SIP server itself is not under a significant load and the Overload Policy has not been activated. This may or may not be a permanent error condition.

17

Upgrading Deployed SIP Applications

The following sections describe how to upgrade deployed SIP Servlets and converged SIP/HTTP applications to a newer version of the same application without losing active calls:

17.1 Overview of SIP Application Upgrades

With Oracle WebLogic Communication Services, you can upgrade a deployed SIP application to a newer version without losing existing calls being processed by the application. This type of application upgrade is accomplished by deploying the newer application version alongside the older version. Oracle WebLogic Communication Services automatically manages the SIP Servlet mapping so that new requests are directed to the new version. Subsequent messages for older, established dialogs are directed to the older application version until the calls complete. After all of the older dialogs have completed and the earlier version of the application is no longer processing calls, you can safely undeploy it.

Oracle WebLogic Communication Services's upgrade feature ensures that no calls are dropped while during the upgrade of a production application. The upgrade process also enables you to revert or rollback the process of upgrading an application. If, for example, you determine that there is a problem with the newer version of the deployed application, you can undeploy the newer version and activate the older version.

Note: When you undeploy an active version of an application, the previous application version remains in administration mode. You must explicitly activate the older version in order to direct new requests to the application.

You can also use the upgrade functionality with a SIP administration channel to deploy a new application version with restricted access for final testing. After performing final testing using the administration channel, you can open the application to general SIP traffic.

Oracle WebLogic Communication Services application upgrades provide the same functionality as Oracle WebLogic Server 10*g* Release 3 application upgrades, with the following exceptions:

- Oracle WebLogic Communication Services does not support "graceful" retirement of old application versions. Instead, only timeout-based undeployment is supported using the `-retiretimeout` option to `weblogic.Deployer`.
- If you want to use administration mode with SIP Servlets or converged applications, you must configure a `sips-admin` channel that uses TLS transport.
- Oracle WebLogic Communication Services handles application upgrades differently in replicated and non-replicated environments. In replicated environments, the server behaves as if the `save-sessions-enabled` element was set to "true" in the `weblogic.xml` configuration file. This preserves sessions across a redeployment operation.

  For non-replicated environments, sessions are destroyed immediately upon redeployment.

17.2 Requirements and Restrictions for Upgrading Deployed Applications

To use the application upgrade functionality of Oracle WebLogic Communication Services:

- You must assign version information to your updated application in order to distinguish it from the older application version. Note that only the newer version of a deployed application requires version information; if the currently-deployed application contains no version designation, Oracle WebLogic Communication Services automatically treats this application as the "older" version. See Section 17.4, "Assign a Version Identifier".
- A maximum of two different versions of the same application can be deployed at one time.
- If your application hard-codes the use of an application name (for example, in composed applications where multiple SIP Servlets process a given call), you must replace the application name with calls to a helper method that obtains the base application name. WebLogic Server provides `ApplicationRuntimeMBean` methods for obtaining the base application name and version identifier, as well as determining whether the current application version is active or retiring. See Section 17.8, "Accessing the Application Name and Version Identifier".
- When applications take part in a composed application (using application composition techniques), Oracle WebLogic Communication Services always uses the latest version of an application when only the base name is supplied.

- If you want to deploy an application in administration mode, you must configure a `sips-admin` channel that uses TLS transport.

17.3 Steps for Upgrading a Deployed SIP Application

Follow these steps to upgrade a deployed SIP application to a newer version:

1. Assign a Version Identifier—Package the updated version of the application with a version identifier.
2. Deploy the Updated Application Version—Deploy the updated version of the application alongside the previous version to initiate the upgrade process.
3. Undeploy the Older Application Version—After the older application has finished processing all SIP messages for its established calls, you can safely undeploy that version. This leaves the newly-deployed application version responsible for processing all current and future calls.

Each procedure is described in the sections that follow. You can also roll back the upgrade process if you discover a problem with the newly-deployed application. Applications that are composed of multiple SIP Servlets may also need to use the `ApplicationRuntimeMBean` for accessing the application name and version identifier.

17.4 Assign a Version Identifier

Oracle WebLogic Communication Services uses a version identifier—a string value—appended to the application name to distinguish between multiple versions of a given application. The version string can be a maximum of 215 characters long, and must consist of valid characters as identified in Table 17–1.

Table 17–1 Valid and Invalid Characters

Valid ASCII Characters	Invalid Version Constructs
a-z	..
A-Z	.
0-9	
period ("."), underscore ("_"), or hyphen ("-") in combination with other characters	

For deployable SIP Servlet WAR files, you must define the version identifier in the MANIFEST.MF file of the application or specify it on the command line at deployment time.

17.4.1 Defining the Version in the Manifest

Both WAR and EAR deployments must specify a version identifier in the MANIFEST.MF file. Example 17–1 shows an application with the version identifier "v2":

Example 17–1 Version Identifier in Manifest

```
Manifest-Version: 1.0
Created-By: 1.4.1_05-b01 (Sun Microsystems Inc.)
Weblogic-Application-Version: v2
```

If you deploy an application without a version identifier, and later deploy with a version identifier, Oracle WebLogic Communication Services recognizes the deployments as separate versions of the same application.

17.5 Deploy the Updated Application Version

To begin the upgrade process, simply deploy the updated application archive using either the Administration Console or `weblogic.Deployer` utility. Use the `-retiretimeout` option to the `weblogic.Deployer` utility if you want to automatically undeploy the older application version after a fixed amount of time. For example:

```
java weblogic.Deployer -name MyApp -version v2 -deploy -retiretimeout 7
```

Oracle WebLogic Communication Services examines the version identifier in the manifest file to determine if another version of the application is currently deployed. If two versions are deployed, the server automatically begins routing new requests to the most recently-deployed application. The server allows the other deployed application to complete in-flight calls, directs no new calls to it. This process is referred to as "retiring" the older application, because eventually the older application version will process no SIP messages.

Note that Oracle WebLogic Communication Services does not compare the actual version strings of two deployed applications to determine which is the higher version. New calls are always routed to the most recently-deployed version of an application.

Oracle WebLogic Communication Services also distinguishes between a deployment that has no version identifier (no version string in the manifest) and a subsequent version that does specify a version identifier. This enables you to easily upgrade applications that were packaged before you began including version information as described in Section 17.4, "Assign a Version Identifier".

17.6 Undeploy the Older Application Version

After deploying a new version of an existing application, the original deployment process messages only for in-flight calls (calls that were initiated with the original deployment). After those in-flight calls complete, the original deployment no longer processes any SIP messages. In most production environments, you will want to ensure that the original deployment is no longer processing messages before you undeploy the application.

To determine whether a deployed application is processing messages, you can obtain the active session count from the application's `SipApplicationRuntimeMBean` instance. Example 17–2 shows the sample WLST commands for viewing the active session count for the `findme` sample application on the default single-server domain.

Based on the active session count value, you can undeploy the application safely (without losing any in-flight calls) or abruptly (losing the active session counts displayed at the time of undeployment).

Use either the Administration Console or `weblogic.Deployer` utility to undeploy the correct deployment name.

Example 17–2 Sample WLST Session for Examining Session Count

```
connect()
custom()
cd ('examples:Location=myserver,Name=myserver_myserver_findme_
```

```
findme,ServerRuntime=myserver,Type=SipApplicationRuntime')
ls()
-rw-   ActiveAppSessionCount                        0
-rw-   ActiveSipSessionCount                        0
-rw-   AppSessionCount                              0
-rw-   CachingDisabled                              true
-rw-   MBeanInfo                                   weblogic.management.tools.In
fo@5ae636
-rw-   Name                                         myserver_myserver_findme_fin
dme
-rw-   ObjectName                                   examples:Location=myserver,N
ame=myserver_myserver_findme_findme,ServerRuntime=myserver,Type=SipApplicationRu
ntime
-rw-   Parent                                       examples:Location=myserver,N
ame=myserver,Type=ServerRuntime
-rw-   Registered                                   false
-rw-   SipSessionCount                              0
-rw-   Type                                         SipApplicationRuntime

-rwx   preDeregister                                void :
```

17.7 Roll Back the Upgrade Process

If you deploy a new version of an application and discover a problem with it, you can roll back the upgrade process by:

1. Undeploying the active version of the application.
2. Activating the older version of the application. For example:

```
java weblogic.Deployer -name MyApp -appversion v1 -start
```

> **Note:** When you undeploy an active version of an application, the previous application version remains in administration mode. You must explicitly activate the older version in order to direct new requests to the application.

Alternately, you can use simply use the `-start` option to start the older application version, which causes the older version of the application to process new requests and retire the newer version.

17.8 Accessing the Application Name and Version Identifier

If you intend to use Oracle WebLogic Communication Services's production upgrade feature, applications that are composed of multiple SIP Servlets should not hard-code the application name. Instead of hard-coding the application name, your application can dynamically access the deployment name or version identifier by using helper methods in `ApplicationRuntimeMBean`.

17.9 Using Administration Mode

You can optionally use the `-adminmode` option with `weblogic.Deployer` to deploy a new version of an application in administration mode. While in administration mode, SIP traffic is accepted only via a configured network channel named `sips-admin` having the TLS transport. If no `sips-admin` channel is configured, or if

a request is received using a different channel, the server rejects the request with a 503 message.

To transition the application from administration mode to a generally-available mode, use the `-start` option with `weblogic.Deployer`.

Note: If using TLS is not feasible with your application, you can alternately change the Servlet role mapping rules to allow only 1 user on the newer version of the application. This enables you to deploy the newer version alongside the older version, while restricting access to the newer version.

18

Parlay X Web Services Architecture

This chapter describes the architecture, security, and installation for the OWLCS Parlay X Web Services. This chapter contains the following sections:

18.1 Architecture of Web Service Client Applications

The architecture of client applications is such that one client of a Web service will be acting on behalf of many end users of the system.

Multiple users can simultaneously connect to the same Web service client, which will act on behalf of those users when invoking the Web Service. Note the following usage guidelines:

- Security – the OWLCS Web server on which the Web services are running authenticates the client and not the end users. The client is a trusted entity and once the client is authenticated it is assumed that all end users of the client are authenticated. This places the task of authenticating end users on the Web service client. For the client to be correctly authenticated, they must connect with pre-determined authentication credentials.
- The Web client need not invoke all the Web services – that is, a web client might invoke methods on the SendMessage Web service only, and therefore has no need to concern itself with the other Web services. However, there are instances where the needs of the client application dictate that it invokes specific methods on the different web services in a specific order to achieve its goals. For example, a client application for sending messages that also wants to receive message notifications for all the messages sent must first call the startMessageNotification method of the MessageNotificationManager interface before it can receive notifications for messages sent.

18.2 Web Service Security

By default, all deployed OWLCS web services are unsecured. Web Service Security should be enabled for any services that will be deployed in a production environment.

OWLCS supports the use of Oracle Web Services Manager WS-Security policies to protect OWLCS web services. For more information about Oracle Web Services Manager, see "Using Oracle Web Service Security Policies", in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*.

The recommended security configuration for OWLCS web services uses Security Assertion Markup Language (SAML) tokens to pass identities between web service clients and OWLCS. With SAML tokens, instead of the web service client passing a username and password to OWLCS, a trust relationship is established between the client and OWLCS by means of exchanging certificates. Once this keystore configuration is in place, the web service client passes only the user identity, and vouches for the fact that it has authenticated the user appropriately.

The recommended policies to use for OWLCS web services are:

- `oracle/wss11_saml_token_with_message_protection_service_policy (server-side)`
- `oracle/wss11_saml_token_with_message_protection_client_policy (client-side)`

18.2.1 Web Service Security on Notification

The different Web services include corresponding notification Web services (MessageNotification, PresenceNotification) that run on the client side and receive notifications (message delivery status, message receipt, presence status change) when the appropriate event occurs. This implementation does not provide for the use of Web Service security (WS-Security) by default during notification of the clients. That is, the server assumes that the notification Web services running on the client side do not use WS-Security, and makes no attempt to authenticate itself when sending notifications. If you enable WS-Security on the client side, the notification from the server will fail because the notification SOAP request will be missing the required headers.

18.2.2 Enabling OWLCS Service Security

To enable a policy for an OWLCS web service, follow the steps in "Configuring Oracle WSM Security Policies in Administration Console" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*, selecting policy `oracle/wss11_saml_token_with_message_protection_service_policy`. This configuration must be repeated for each service that you wish to secure.

18.2.3 Enabling Client Security

Web service client security must be enabled programmatically. When using the client libraries described in *Oracle WebLogic Communication Services Developer's Guide*, WS-Security policy configuration is provided when a client object is constructed. The client constructors take an argument of type `Map<String, Object>`. In general when using SAML authentication, the key/value pairs (Table 18–1) should be added to the configuration map in addition to other required properties such as the endpoint address.

Table 18–1 Client security keys

Key	Type	Typical Value
`oracle.sdp.parlayx.ParlayXConstants.POLICIES`	String[]	`oracle/wss11_saml_token_with_message_protection_client_policy`
`javax.xml.ws.BindingProvider.USERNAME_PROPERTY`	String	*`<valid username>`*

Table 18–1 (Cont.) Client security keys

Key	Type	Typical Value
`oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_ALIAS_PROPERTY`	String	(optional) keystore alias for target service. See Client Aliases.

Example 18–1 Web Service Client Security

```
import oracle.sdp.parlayx.presence.consumer.PresenceConsumerClient;

...

Map<String, Object> config = new HashMap<String, Object>();
config.put(javax.xml.ws.BindingProvider.ENDPOINT_ADDRESS_PROPERTY, owlcs_url);
config.put(oracle.sdp.parlayx.ParlayXConstants.POLICIES, new String[]
{"oracle/wss11_saml_token_with_message_protection_client_policy"});
config.put(javax.xml.ws.BindingProvider.USERNAME_PROPERTY, "test.user1");

PresenceConsumerClient presenceClient = new PresenceConsumerClient(config);
```

18.2.4 Keystore Configuration

In order to use the recommended WS-Security policy, you must configure a keystore containing the public and private key information required by OWSM. Refer to "Configuring the Credential Store Using WLST" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server* for information on how to configure the keystore and corresponding credential store entries.

- If both your web service client and OWLCS server are in the same domain, then they share a keystore and credential store.
- If your web service client and OWLCS server are in different domains, then you must import the OWLCS public key into your client domain's keystore, and must import your client domain's public key into the OWLCS keystore.

18.2.5 Client Aliases

When using certain WS-Security policies such as the SAML policy recommended here, the client must use the server's public key to encrypt the web service request. However, there is generally only one keystore configured per domain. Therefore, if you have a domain in which there are web service clients that communicate with web services in multiple other domains, then you may need to override the default keystore entry used by OWSM.

For example, if you have a domain in which application "A" is a web service client to a SOA web service, and application "B" is a web service client to an OWLCS web service, then A's requests must be encrypted using the public key of the SOA domain, and B's requests must be encrypted using the public key of the OWLCS domain. You can accomplish this goal by overriding the keystore alias used by OWSM for each request.

- Import the two server domain's public keys into the client domain's keystore using different keystore aliases. For example, import the OWLCS public key with alias "owlcs_public_key", and the SOA public key with alias "soa_public_key".
- When creating an OWLCS web service client, specify the recipient keystore alias parameter, setting the key to `oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_`

RECIPIENT_ALIAS_PROPERTY and the value to "owlcs_public_key" as shown in Example 18–2.

Example 18–2 Client Aliases

```
import oracle.sdp.parlayx.presence.consumer.PresenceConsumerClient;

...

Map<String, Object> config = new HashMap<String, Object>();
config.put(javax.xml.ws.BindingProvider.ENDPOINT_ADDRESS_PROPERTY, owlcs_url);
config.put(oracle.sdp.parlayx.ParlayXConstants.POLICIES, new String[]
{"oracle/wss11_saml_token_with_message_protection_client_policy"});
config.put(javax.xml.ws.BindingProvider.USERNAME_PROPERTY, "test.user1");
config.put(oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_
ALIAS_PROPERTY, "owlcs_public_key")
PresenceConsumerClient presenceClient = new PresenceConsumerClient(config);
```

- The SOA or other web service client will similarly need to override the keystore alias, but the exact mechanism may differ. For example if using a JAX-WS client stub directly, then you can add the override property to the JAX-WS request context. See "Policy Configuration Overrides for the Web Service Client" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server* for more details.

18.3 Installing the Web Services

The Web services are packaged as a standard .ear file and can be deployed the same as any other Web services through Enterprise Manager. The .ear file contains two .war files that implement the two interfaces. The web services are dependent on the following shared libraries: `oracle.sdp.client`, `oracle.sdp.platform`, `oracle.sdp.presencecommons`.

Your client applications need to import (and be compiled against) the oracle.sdp.client shared library that is provided with OWLCS. This consists of importing parlayx.jar into your projects.

In addition to compiling against the oracle.sdp.client shared library, this shared library should also be available in the target runtime environment (that is, oracle.sdp.client is deployed as a shared library in your target Weblogic Server).

In addition to the shared library above, the OWLCS installation contains war files for the notification Web services. These war files contain all the necessary jar files that developers need to import to enable notification for the different Web services:

- messagingwsnotification-<version>.war – deployable war file that contains jars that should be imported when building a client intends to receive notifications for message delivery status and message reception. This war should also be deployed along with the client application in order for the OWLCS server to be able to invoke the messaging notification Web service.
- presencewsnotification-<version>.war – deployable war file that contains jars that should be imported when building a client intends to receive notifications for presence status changes. This war should also be deployed along with the client application so that the OWLCS server can invoke the presence notification Web service.

Part VII

Reference

This Part contains the following appendices:

- Appendix A, "SIP Servlet Container Configuration Reference"
- Appendix B, "SIP Data Tier Configuration Reference"
- Appendix C, "Diameter Configuration Reference"
- Appendix D, "Startup Command Options"
- Appendix E, "Supported Platforms, Protocols, RFCs and Standards"

A

SIP Servlet Container Configuration Reference

The following sections provide a complete reference to the engine tier configuration file, `sipserver.xml`:

- Section A.1, "Overview of sipserver.xml"
- Section A.2, "Editing sipserver.xml"
- Section A.3, "XML Schema"
- Section A.4, "Example sipserver.xml File"
- Section A.5, "XML Element Description"

A.1 Overview of sipserver.xml

The `sipserver.xml` file is an XML document that configures the SIP container features provided by an Oracle WebLogic Communication Services instance in the engine tier of a server installation. `sipserver.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where *`DOMAIN_DIR`* is the root directory of the Oracle WebLogic Communication Services domain.

A.2 Editing sipserver.xml

You should never move, modify, or delete the `sipserver.xml` file during normal operations.

Oracle recommends using the Administration Console to modify `sipserver.xml` indirectly, rather than editing the file by hand. Using the Administration Console ensures that the `sipserver.xml` document always contains valid XML. See also Configuring Container Properties Using WLST (JMX) in the *Configuration Guide*.

You may need to manually view or edit `sipserver.xml` to troubleshoot problem configurations, repair corrupted files, or to roll out custom configurations to a large number of machines when installing or upgrading Oracle WebLogic Communication Services. When you manually edit `sipserver.xml`, you must reboot Oracle WebLogic Communication Services instances to apply your changes.

Caution: Always use the SipServer node in the Administration Console or the WLST utility to make changes to a running Oracle WebLogic Communication Services deployment.

A.2.1 Steps for Editing sipserver.xml

If you need to modify `sipserver.xml` on a production system, follow these steps:

1. Use a text editor to open the *DOMAIN_DIR*`/config/custom/sipserver.xml` *file, where DOMAIN_DIR* is the root directory of the Oracle WebLogic Communication Services domain.
2. Modify the `sipserver.xml` file as necessary. See Section A.3, "XML Schema" for a full description of the XML elements.
3. Save your changes and exit the text editor.
4. Reboot or start servers to have your changes take effect:

   Caution: Always use the SipServer node in the Administration Console or the WLST utility to make changes to a running Oracle WebLogic Communication Services deployment.

5. Test the updated system to validate the configuration.

A.3 XML Schema

The schema file for `sipserver.xml`, `wcp-sipserver.xsd`, is installed inside the `wlss-descriptor-binding.jar` library, located in the *WLSS_HOME*`/server/lib/wlss` directory.

A.4 Example sipserver.xml File

The following shows a simple example of a `sipserver.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sip-server xmlns="http://www.bea.com/ns/wlcp/wlss/300">
  <overload>
    <threshold-policy>queue-length</threshold-policy>
    <threshold-value>200</threshold-value>
    <release-value>150</release-value>
  </overload>
</sip-server>
```

A.5 XML Element Description

The following sections describe each element used in the `sipserver.xml` configuration file. Each section describes an XML element that is contained within the main `sip-server` element shown in Figure A–1.

A.5.1 enable-timer-affinity

The `enable-timer-affinity` element determines the way in which engine tier servers process expired timers. By default (when `enable-timer-affinity` is omitted from `sipserver.xml`, or is set to "false"), an engine tier server that polls the SIP data tier for expired timers processes all available expired timers. When `enable-timer-affinity` is set to "true," engine tier servers polling the SIP data tier process only those expired timers that are associated with call states that the engine last modified (or expired timers for call states that have no owner).

A.5.2 overload

The `overload` element enables you to throttle incoming SIP requests according to a configured overload condition. When an overload condition occurs, Oracle WebLogic Communication Services destroys new SIP requests by responding with "503 Service Unavailable" until the configured release value is observed, or until the size of the server's capacity constraints is reduced (see Section A.5.2.3, "Overload Control Based on Capacity Constraints").

User-configured overload controls are applied only to initial SIP requests; SIP dialogues that are already active when an overload condition occurs may generate additional SIP requests that are not throttled.

To configure an overload control, you define the three elements described in Table A–1.

Table A–1 Nested overload Elements

Element	Description
`threshold-policy`	A String value that identifies the type of measurement used to monitor overload conditions: ▪ `session-rate` measures the rate at which new SIP requests are generated. Oracle WebLogic Communication Services determines the session rate by calculating the number of new SIP application connections that were created in the last 5 seconds of operation. See Section A.5.2.2, "Overload Control Based on Session Generation Rate". ▪ `queue-length` measures the sum of the sizes of the capacity constraint work manager components that processes SIP requests and SIP timers. See Section A.5.2.3, "Overload Control Based on Capacity Constraints". ▪ **Note:** Execute queues are deprecated and no longer used in `Oracle WebLogic Communication Services. Capacity constraints are used in place of execute queues. The policy name "queue-length"` was kept for backward compatibility. You must use only one of the above policies to define an overload control. See Section A.5.2.1, "Selecting an Appropriate Overload Policy" for more information.

Table A–1 (Continued)Nested overload Elements

Element	Description
`threshold-value`	Specifies the measured value that causes Oracle WebLogic Communication Services to recognize an overload condition and *start* throttling new SIP requests: ■ When using the `session-rate` threshold policy, `threshold-value` specifies the number of new SIP requests per second that trigger an overload condition. See Section A.5.2.2, "Overload Control Based on Session Generation Rate". ■ When using the `queue-length` threshold policy, `threshold-value` specifies the size of the combined number of requests in the SIP transport and SIP timer capacity constraint components that triggers an overload condition. See Section A.5.2.3, "Overload Control Based on Capacity Constraints". ■ After the `threshold-value` is observed, Oracle WebLogic Communication Services recognizes an overload condition for a minimum of 512 milliseconds during which time new SIP requests are throttled. If multiple overloads occur over a short period of time, the minimum overload of 512 ms is dynamically increased to avoid repeated overloads. ■ After the minimum overload recognition period expires, the overload condition is terminated only after the configured `release-value` is observed.
`release-value`	Specifies the measured value that causes Oracle WebLogic Communication Services to end an overload condition and *stop* throttling new SIP requests: ■ When using the `session-rate` threshold policy, `release-value` specifies the number of new SIP requests per second that terminates session throttling. See Section A.5.2.2, "Overload Control Based on Session Generation Rate". ■ When using the `queue-length` threshold policy, `release-value` specifies the combined number of requests in the capacity constraints that terminates session throttling. See Section A.5.2.3, "Overload Control Based on Capacity Constraints".

A.5.2.1 Selecting an Appropriate Overload Policy

Oracle WebLogic Communication Services provides two different policies for throttling SIP requests:

- The `session-rate` policy throttles sessions when the volume new SIP sessions reaches a configured rate (a specified number of sessions per second).
- The `queue-length` policy throttles requests after the sum of the requests in the `wlss.connect` work manager and `wlss.timer.capacity` capacity constraint components reaches a configured size.

Note that you must select only one of the available overload policies. You cannot use both policies simultaneously.

The `session-rate` policy is generally used when a back-end resource having a known maximum throughput (for example, an RDBMS) is used to set up SIP calls. In this case, the `session-rate` policy enables you to tie the Oracle WebLogic Communication Services overload policy to the known throughput capabilities of the back-end resource.

With the `queue-length` policy, Oracle WebLogic Communication Services monitors both CPU and I/O bottlenecks to diagnose an overload condition. The `queue-length` policy is generally used with CPU-intensive SIP applications in systems that have no predictable upper bound associated with the call rate.

The following sections describe each policy in detail.

A.5.2.2 Overload Control Based on Session Generation Rate

Oracle WebLogic Communication Services calculates the session generation rate (sessions per second) by monitoring the number of application sessions created in the last 5 seconds. When the session generation rate exceeds the rate specified in the `threshold-value` element, Oracle WebLogic Communication Services throttles initial SIP requests until the session generation rate becomes smaller than the configured `release-value`.

The following example configures Oracle WebLogic Communication Services to begin throttling SIP requests when the new sessions are created at a rate higher than 50 sessions per second. Throttling is discontinued when the session rate drops to 40 sessions per second:

```
<overload>
  <threshold-policy>session-rate</threshold-policy>
  <threshold-value>50</threshold-value>
  <release-value>40</release-value>
</overload>
```

A.5.2.3 Overload Control Based on Capacity Constraints

`By default, SIP messages are handled by a work manager named wlss.connect and SIP timers are processed by a work manager named wlss.timer`.Each work manager has an associated capacity constraint component that sets the number of requests allotted for SIP message handling and timer processing. Work managers are configured in the config.xml file for your Oracle WebLogic Communication Services. You can also allocate additional threads to the server at boot time using the startup option `-Dweblogic.threadpool.MinPoolSize=number_of_threads`.

Oracle WebLogic Communication Services performs `queue-length` overload control by monitoring the combined lengths of the configured capacity constraints. When the sum of the requests in the two constraints exceeds the length specified in the `threshold-value` element, Oracle WebLogic Communication Services throttles initial SIP requests until the total requests are reduced to the configured `release-value`.

Example A–1 shows a sample `overload` configuration from `sipserver.xml`. Here, Oracle WebLogic Communication Services begins throttling SIP requests when the combined size of the constraints exceeds 200 requests. Throttling is discontinued when the combined length returns to 200 or fewer simultaneous requests.

Example A–1 Sample overload Definition

```
<overload>
  <threshold-policy>queue-length</threshold-policy>
  <threshold-value>200</threshold-value>
  <release-value>150</release-value>
</overload>
```

A.5.2.4 Two Levels of Overload Protection

User-configured overload controls (defined in `sipserver.xml`) represent the first level of overload protection provided by Oracle WebLogic Communication Services. They mark the onset of an overload condition and initiate simple measures to avoid dropped calls (generating 503 responses for new requests).

If the condition that caused the overload persists or worsens, then the work manager component used to perform work in the SIP Servlet container may itself become overloaded. At this point, the server no longer utilizes threads to generate 503 responses, but instead begins to drop messages. In this way, the configured size of the SIP container's work manager components represent the second and final level of overload protection employed by the server.

Always configure overload controls in `sipserver.xml` conservatively, and resolve the circumstances that caused the overload in a timely fashion.

A.5.3 message-debug

The `message-debug` element is used to enable and configure access logging with log rotation for Oracle WebLogic Communication Services. This element should be used only in a development environment, because access logging logs *all* SIP requests and responses.

A.5.4 proxy—Setting Up an Outbound Proxy Server

RFC 3261 defines an outbound proxy as "A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols."

In Oracle WebLogic Communication Services an outbound proxy server is specified using the `proxy` element in `sipserver.xml`. The proxy element defines one or more proxy server URIs. You can change the behavior of the proxy process by setting a proxy policy with the `proxy-policy` tag. Example A–1 describes the possible values for the `proxy` elements.

The default behavior is as if **proxy** policy is in effect. The **proxy** policy means that the request is sent out to the configured outbound proxy and the route headers in the request preserve any routing decision taken by Oracle WebLogic Communication Services. This enables the outbound proxy to send the request over to the intended recipient after it has performed its actions on the request. The **proxy** policy comes into effect only for the initial requests. As for the subsequent request the Route Set takes precedence over any policy in a dialog. (If the outbound proxy wants to be in the Route Set it can turn record routing on).

Also if a proxy application written on Oracle WebLogic Communication Services wishes to override the configured behavior of outbound proxy traversal, then it can add a special header with name "X-BEA-Proxy-Policy" and value "domain". This header is stripped from the request while sending, but the effect is to ignore the configured outbound proxy. The X-BEA-Proxy-Policy custom header can be used by applications to override the configured policy on a request-by-request basis. The value of the header can be "domain" or "proxy". Note, however, that if the policy is overridden to "proxy," the configuration must still have the outbound proxy URIs in order to route to the outbound proxy.

Table A–2 Nested proxy Elements

Element	Description
routing-policy	An optional element that configures the behavior of the proxy. Valid values are: ■ **domain** - Proxies messages using the routing rule defined by RFC 3261, ignoring any outbound proxy that is specified. ■ **proxy** - Sends the message to the downstream proxy specified in the default proxy URI. If there are multiple proxy specifications they are tried in the order in which they are specified. However, if the transport tries a UDP proxy, the settings for subsequent proxies are ignored.
uri	The TCP or UDP URI of the proxy server. You must specify at least one URI for a proxy element. Place multiple URIs in multiple uri elements within the proxy element.

Example A–2 shows the default proxy configuration for Oracle WebLogic Communication Services domains. The request in this case is created in accordance with the SIP routing rules, and finally the request is sent to the outbound proxy "sipoutbound.oracle.com".

Example A–2 Sample proxy Definition

```
<proxy>
     <routing-policy>proxy</routing-policy>
     <uri>sip:sipoutbound.oracle.com:5060</uri>
     <!-- Other proxy uri tags can be added. - >
</proxy>
```

A.5.5 t1-timeout-interval

This element sets the value of the SIP protocol T1 timer, in milliseconds. Timer T1 also specifies the initial values of Timers A, E, and G, which control the retransmit interval for INVITE requests and responses over UDP.

Timer T1 also affects the values of timers F, H, and J, which control retransmit intervals for INVITE responses and requests; these timers are set to a value of 64*T1 milliseconds. See the *SIP: Session Initiation Protocol* for more information about SIP timers.

If t1-timeout-interval is not configured, Oracle WebLogic Communication Services uses the SIP protocol default value of 500 milliseconds.

A.5.6 t2-timeout-interval

This elements sets the value of the SIP protocol T2 timer, in milliseconds. Timer T2 defines the retransmit interval for INVITE responses and non-INVITE requests. See the *SIP: Session Initiation Protocol* for more information about SIP timers.

If t2-timeout-interval is not configured, Oracle WebLogic Communication Services uses the SIP protocol default value of 4 seconds.

A.5.7 t4-timeout-interval

This elements sets the value of the SIP protocol T4 timer, in milliseconds. Timer T4 specifies the maximum length of time that a message remains in the network. Timer T4 also specifies the initial values of Timers I and K, which control the wait times for retransmitting ACKs and responses over UDP.

If t4-timeout-interval is not configured, Oracle WebLogic Communication Services uses the SIP protocol default value of 5 seconds.

A.5.8 timer-b-timeout-interval

This elements sets the value of the SIP protocol Timer B, in milliseconds. Timer B specifies the length of time a client transaction attempts to retry sending a request.

If timer-b-timeout-interval is not configured, the Timer B value is derived from timer T1 (64*T1, or 32000 milliseconds by default).

A.5.9 timer-f-timeout-interval

This elements sets the value of the SIP protocol Timer F, in milliseconds. Timer F specifies the timeout interval for retransmitting non-INVITE requests.

If timer-f-timeout-interval is not configured, the Timer F value is derived from timer T1 (64*T1, or 32000 milliseconds by default).

A.5.10 max-application-session-lifetime

This element sets the maximum amount of time, in minutes, that a SIP application session can exist before Oracle WebLogic Communication Services invalidates the session. max-application-session-lifetime acts as an upper bound for any timeout value specified using the session-timeout element in a sip.xml file, or using the setExpires API.

A value of -1 (the default) specifies that there is no upper bound to application-configured timeout values.

A.5.11 enable-local-dispatch

enable-local-dispatch is a server optimization that helps avoid unnecessary network traffic when sending and forwarding messages. You enable the optimization by setting this element "true." When enable-local-dispatch enabled, if a server instance needs to send or forward a message and the message destination is the engine tier's cluster address or the local server address, then the message is routed internally to the local server instead of being sent via the network. Using this optimization can dramatically improve performance when chained applications process the same request.

You may want to disable this optimization if you feel that routing internal messages could skew the load on servers in the engine tier, and you prefer to route all requests via a configured load balancer.

By default enable-local-dispatch is set to "false."

A.5.12 cluster-loadbalancer-map

The cluster-loadbalancer-map element is used only when upgrading Oracle WebLogic Communication Services software, or when upgrading a production SIP Servlet to a new version. It is not required or used during normal server operations.

During a software upgrade, multiple engine tier clusters are defined to host the older and newer software versions. A cluster-loadbalancer-map defines the virtual IP address (defined on your load balancer) that correspond to an engine tier cluster configured for an upgrade. Oracle WebLogic Communication Services uses this mapping to ensure that engine tier requests for timers and call state data are received from the correct "version" of the cluster. If a request comes from an incorrect version of

the software, the `cluster-loadbalancer-map` entries are used to forward the request to the correct cluster.

Each `cluster-loadbalancer-map` entry contains the two elements described in Table A–1.

Table A–3 Nested cluster-loadbalancer-map Elements

Element	Description
`cluster-name`	The configured name of an engine tier cluster.
`sip-uri`	The internal SIP URI that maps to the engine tier cluster. This corresponds to a virtual IP address that you have configured in your load balancer. The internal URI is used to forward requests to the correct cluster version during an upgrade.

Example A–3 shows a sample `cluster-loadbalancer-map` entry used during an upgrade.

Example A–3 Sample cluster-loadbalancer-map Entry

```
<cluster-loadbalancer-map>
   <cluster-name>EngineCluster</cluster-name>
   <sip-uri>sip:172.17.0.1:5060</sip-uri>
</cluster-loadbalancer-map>
<cluster-loadbalancer-map>
   <cluster-name>EngineCluster2</cluster-name>
   <sip-uri>sip:172.17.0.2:5060</sip-uri>
</cluster-loadbalancer-map>
```

See Upgrading Software in the Operations Guide for more information.

A.5.13 default-behavior

This element defines the default behavior of the Oracle WebLogic Communication Services instance if the server cannot match an incoming SIP request to a deployed SIP Servlet (or if the matching application has been invalidated or timed out). Valid values are:

- `proxy`—Act as a proxy server.
- `ua`—Act as a User Agent.

`proxy` is used as the default if you do not specify a value.

When acting as a User Agent (UA), Oracle WebLogic Communication Services acts in the following way in response to SIP requests:

- ACK requests are discarded without notice.
- CANCEL or BYE requests receive response code 481 - Transaction does not exist.
- All other requests receive response code 500 - Internal server error.

When acting as a proxy requests are automatically forwarded to an outbound proxy (see Section A.5.4, "proxy—Setting Up an Outbound Proxy Server") if one is configured. If no proxy is defined, Oracle WebLogic Communication Services proxies to a specified Request URI only if the Request URI does not match the IP and port number of a known local address for a SIP Servlet container, or a load balancer address configured for the server. This ensures that the request does not constantly loop to the same servers. When the Request URI matches a local container address or

load balancer address, Oracle WebLogic Communication Services instead acts as a UA.

A.5.14 default-servlet-name

This element specifies the name of a default SIP Servlet to call if an incoming initial request cannot be matched to a deployed Servlet (using standard `servlet-mapping` definitions in `sip.xml`). The name specified in the `default-servlet-name` element must match the `servlet-name` value of a deployed SIP Servlet. For example:

```
<default-servlet-name>myServlet</default-servlet-name>
```

If the name defined in `default-servlet-name` does not match a deployed Servlet, or no value is supplied (the default configuration), Oracle WebLogic Communication Services registers the name `com.bea.wcp.sip.engine.BlankServlet` as the default Servlet. The `BlankServlet` name is also used if a deployed Servlet registered as the `default-servlet-name` is undeployed from the container.

`BlankServlet`'s behavior is configured with the `default-behavior` element. By default the Servlet proxies all unmatched requests. However, if the `default-behavior` element is set to "ua" mode, `BlankServlet` is responsible for returning 481 responses for CANCEL and BYE requests, and 500/416 responses in all other cases. `BlankServlet` does not respond to ACK, and it always invalidates the application session.

A.5.15 retry-after-value

Specifies the number of seconds used in the `Retry-After` header for 5xx responses. This value can also include a parameter or a reason code, such as "Retry-After: 18000;duration=3600" or "Retry-After: 120 (I'm in a meeting)."

If the this value is not configured, Oracle WebLogic Communication Services uses the default value of 180 seconds.

A.5.16 sip-security

Oracle WebLogic Communication Services enables you to configure one or more trusted hosts for which authentication is not performed. When Oracle WebLogic Communication Services receives a SIP message, it calls `getRemoteAddress()` on the SIP Servlet message. If this address matches an address defined in the server's trusted host list, no further authentication is performed for the message.

The `sip-security` element defines one or more trusted hosts, for which authentication is not performed. The `sip-security` element contains one or more `trusted-authentication-host` or `trusted-charging-host` elements, each of which contains a trusted host definition. A trusted host definition can consist of an IP address (with or without wildcard placeholders) or a DNS name. Example A–4 shows a sample `sip-security` configuration.

Example A–4 Sample Trusted Host Configuration

```
<sip-security>

<trusted-authentication-host>myhost1.mycompany.com</trusted-authentication-host>
   <trusted-authentication-host>172.*</trusted-authentication-host>
</sip-security>
```

A.5.17 route-header

3GPP TS 24.229 Version 7.0.0 (http://www.3gpp.org/ftp/Specs/archive/24_series/24.229/24229-700.zip) requires that IMS Application Servers generating new requests (for example, as a B2BUA) include the S-CSCF route header. In Oracle WebLogic Communication Services, the S-CSCF route header must be statically defined as the value of the `route-header` element in `sipserver.xml`. For example:

```
<route-header>
   <uri>Route: sip:wlss1.bea.com</uri>
</route-header>
```

A.5.18 engine-call-state-cache-enabled

Oracle WebLogic Communication Services provides the option for engine tier servers to cache a portion of the call state data locally, as well as in the SIP data tier, to improve performance with SIP-aware load balancers. When a local cache is used, an engine tier server first checks its local cache for existing call state data. If the cache contains the required data, and the local copy of the data is up-to-date (compared to the SIP data tier copy), the engine locks the call state in the SIP data tier but reads directly from its cache.

By default the engine tier cache is enabled. To disable caching, set `engine-call-state-cache-enabled` to false:

```
<engine-call-state-cache-enabled>false</engine-call-state-cache-enabled>
```

A.5.19 server-header

Oracle WebLogic Communication Services enables you to control when a Server header is inserted into SIP messages. You can use this functionality to limit or eliminate Server headers to reduce the message size for wireless networks, or to increase security.

By default, Oracle WebLogic Communication Services inserts no Server header into SIP messages. Set the `server-header` to one of the following string values to configure this behavior:

- `none` (the default) inserts no Server header.
- `request` inserts the Server header only for SIP requests generated by the server.
- `response` inserts the Server header only for SIP responses generated by the server.
- `all` inserts the Server header for all SIP requests and responses.

For example, the following element configures Oracle WebLogic Communication Services to insert a Server header for all generated SIP messages:

```
<server-header>all</server-header>
```

See also Section A.5.20, "server-header-value".

A.5.20 server-header-value

Oracle WebLogic Communication Services enables you to control the text that is inserted into the Server header of generated messages. This provides additional control over the size of SIP messages and also enables you to mask the server entity for security purposes. By default, Oracle WebLogic Communication Services does not

insert a Server header into generated SIP messages (see Section A.5.19, "server-header"). If Server header insertion is enabled but no `server-header-value` is specified, Oracle WebLogic Communication Services inserts the value "WebLogic SIP Server." To configure the header contents, enter a string value. For example:

```
<server-header-value>MyCompany Application Server</server-header-value>
```

A.5.21 persistence

The `persistence` element defines enables or disables writing call state data to an RDBMS and/or to a remote, geographically-redundant Oracle WebLogic Communication Services installation. For sites that utilize geographically-redundant replication features, the `persistence` element also defines the site ID and the URL at which to persist call state data.

The `persistence` element contains the sub-elements described in Table A–1.

Table A–4 Nested persistence Elements

Element	Description
`default-handling`	Determines whether or not Oracle WebLogic Communication Services observes persistence hints for RDBMS persistence and/or geographical-redundancy. This element can have one of the following values: ▪ **all**—Specifies that call state data may be persisted to both an RDBMS store and to a geographically-redundant Oracle WebLogic Communication Services installation. This is the default behavior. Note that actual replication to either destination also requires that the available resources (JDBC datasource and remote JMS queue) are available. ▪ **db**—Specifies that long-lived call state data is replicated to an RDBMS if the required JDBC datasource and schema are available. ▪ **geo**—Specifies that call state data is persisted to a remote, geographically-redundant site if the configured site URL contains the necessary JMS resources. ▪ **none**—Specifies that only in-memory replication is performed to other replicas in the SIP data tier cluster. Call state data is not persisted in an RDBMS or to an external site.
`geo-site-id`	Specifies the site ID of this installation. All installations that participate in geographically-redundant replication require a unique site ID.
`geo-remote-t3-url`	Specifies the remote Oracle WebLogic Communication Services installation to which this site replicates call state data. You can specify a single URL corresponding to the engine tier cluster of the remote installation. You can also specify a comma-separated list of addresses corresponding to each engine tier server. The URLs must specify the t3 protocol.

Example A–5 shows a sample configuration that uses RDBMS storage for long-lived call state as well as geographically-redundant replication. Call states are replicated to two engine tier servers in a remote location.

Example A–5 Sample persistence Configuration

```
<persistence>
  <default-handling>all</default-handling>
  <geo-site-id>1</geo-site-id>

<geo-remote-t3-url>t3://remoteEngine1:7050,t3://remoteEngine2:7051</geo-remote-t3-
```

```
url>
</persistence>
```

A.5.22 use-header-form

This element configures the server-wide, default behavior for using or preserving compact headers in SIP messages. You can set this element to one of the following values:

- `compact`—Oracle WebLogic Communication Services uses the compact form for all system-generated headers. However, any headers that are copied from an originating message (rather than generated) use their original form.
- `force compact`—Oracle WebLogic Communication Services uses the compact form for all headers, converting long headers in existing messages into compact headers as necessary.
- `long`—Oracle WebLogic Communication Services uses the long form for all system-generated headers. However, any headers that are copied from an originating message (rather than generated) use their original form.
- `force long`—Oracle WebLogic Communication Services uses the long form for all headers, converting compact headers in existing messages into long headers as necessary.

A.5.23 enable-dns-srv-lookup

This element enables or disables Oracle WebLogic Communication Services DNS lookup capabilities. If you set the element to "true," then the server can use DNS to:

- Discover a proxy server's transport, IP address, and port number when a request is sent to a SIP URI.
- Resolve an IP address and/or port number during response routing, depending on the contents of the Sent-by field.

For proxy discovery, Oracle WebLogic Communication Services uses DNS resolution only once per SIP transaction to determine transport, IP, and port number information. All retransmissions, ACKs, or CANCEL requests are delivered to the same address and port using the same transport. For details about how DNS resolution takes place, see *RFC 3263: Session Initiation Protocol (SIP): Locating SIP Servers* (`http://www.ietf.org/rfc/rfc3263.txt`).

When a proxy needs to send a response message, Oracle WebLogic Communication Services uses DNS lookup to determine the IP address and/or port number of the destination, depending on the information provided in the sent-by field and via header.

By default, DNS resolution is not used ("false").

> **Note:** Because DNS resolution is performed within the context of SIP message processing, any DNS performance problems result in increased latency performance. Oracle recommends using a caching DNS server in a production environment to minimize potential performance problems.

A.5.24 connection-reuse-pool

Oracle WebLogic Communication Services includes a connection pooling mechanism that can be used to minimize communication overhead with a Session Border Control (SBC) function or Serving Call Session Control Function (S-CSCF). You can configure multiple, fixed pools of connections to different addresses.

Oracle WebLogic Communication Services opens new connections from the connection pool on demand as the server makes requests to a configured address. The server then multiplexes new SIP requests to the address using the already-opened connections, rather than repeatedly terminating and recreating new connections. Opened connections are re-used in a round-robin fashion. Opened connections remain open until they are explicitly closed by the remote address.

Note that connection re-use pools are not used for incoming requests from a configured address.

To configure a connection re-use pool, you define the four nested elements described in Table A–1.

Table A–5 Nested connection-reuse-pool Elements

Element	Description
`pool-name`	A String value that identifies the name of this pool. All configured `pool-name` elements must be unique to the domain.
`destination`	Specifies the IP address or host name of the destination SBC or S-CSCF. Oracle WebLogic Communication Services opens or re-uses connection in this pool only when making requests to the configured address.
`destination-port`	Specifies the port number of the destination SBC or S-CSCF.
`maximum-connections`	Specifies the maximum number of opened connections to maintain in this pool.

Example A–6 shows a sample `connection-reuse-pool` configuration having two pools.

Example A–6 Sample connection-reuse-pool Configuration

```
<connection-reuse-pool>
   <pool-name>SBCPool</pool-name>
   <destination>MySBC</destination>
   <destination-port>7070</destination-port>
   <maximum-connections>10</maximum-connections>
</connection-reuse-pool>
<connection-reuse-pool>
   <pool-name>SCSFPool</pool-name>
   <destination>192.168.1.6</destination>
   <destination-port>7071</destination-port>
   <maximum-connections>10</maximum-connections>
</connection-reuse-pool>
```

A.5.25 globally-routable-uri

This element enables you to specify a Globally-Routable User Agent URI (GRUU) that Oracle WebLogic Communication Services automatically inserts into Contact and Route-Set headers when communicating with network elements. The URI specified in this element should be the GRUU for the entire Oracle WebLogic Communication Services cluster. (In a single-server domain, use a GRUU for the server itself.)

Note that User Agents (UAs) deployed on Oracle WebLogic Communication Services typically obtain GRUUs via a registration request. In this case, the application code is responsible both for requesting and subsequently handling the GRUU. To request a GRUU the UA would include the "+sip.instance" Contact header field parameter in each Contact for which GRUU is required. Upon receiving a GRUU, the UA would use the GRUU as the URI for the contact header field when generating new requests.

A.5.26 domain-alias-name

This element defines one or more domains for which Oracle WebLogic Communication Services is responsible. If a message has a destination domain that matches a domain specified with a `domain-alias-name` element, Oracle WebLogic Communication Services processes the message locally, rather than forwarding it.

The `sipserver.xml` configuration file can have multiple `main-alias-name` elements. Each element can specify either:

- an individual, fully-qualified domain name, such as `myserver.mycompany.com`, or
- a domain name starting with an initial wildcard character, such as `*.mycompany.com`, used to represent all matching domains. Note that only a single wildcard character is supported, and it must be used as the first element of the domain name.

> **Note:** You can also identify these domain names using the Domain Aliases field in the Configuration > General tab of the SipServer Administration Console extension.

A.5.27 enable-rport

This element determines whether or not Oracle WebLogic Communication Services automatically adds an `rport` parameter to `Via` headers when acting as a UAC. By default, the server does not add the `rport` parameter; set the element to "true" to automatically add `rport` to requests generated by the server.

> **Note:** You can also set this parameter to "true" by selecting the Symmetric Response Routing option on the Configuration > General tab of the SipServer Administration console extension.

The `rport` parameter is used for symmetric response routing as described in RFC 3581 (`http://www.ietf.org/rfc/rfc3581.txt`). When a message is received by an RFC 3581-compliant server, such as Oracle WebLogic Communication Services, the server responds using the remote UDP port number from which the message was received, rather than the port number specified in the `Via` header. This behavior is frequently used when servers reside behind gateway devices that perform Network Address Translation (NAT). The NAT devices maintain a binding between the internal and external port numbers, and all communication must be initiated via the gateway port.

Note that Oracle WebLogic Communication Services is compliant with RFC 3581, and will honor the `rport` parameter even if you set the `enable-rport` element to "false." The `enable-rport` element only specifies whether the server automatically adds `rport` to the requests it generates when acting as a UAC. To disable `rport` handling

completely (disable RFC 3581 support), you must start the server with the command-line option, `-Dwlss.udp.uas.rport=false`.

> **Note:** `rport` support as described in RFC 3581 requires that SIP responses include the source port of the original SIP request. Because source port information is frequently treated as sensitive data, Oracle recommends using the TLS transport.

A.5.28 image-dump-level

This element specifies the level of detail to record in Oracle WebLogic Communication Services diagnostic image files. You can set this element to one of two values:

- `basic`—Records all diagnostic data except for call state data.
- `full`—Records all diagnostic data including call state data.

> **Note:** Recording call state data in the image file can be time consuming. By default, image dump files are recorded using the `basic` option.
>
> You can also set this parameter using the Configuration > General tab of the SipServer Administration console extension.

A.5.29 stale-session-handling

Oracle WebLogic Communication Services uses encoded URIs to identify the call states and application sessions associated with a message. When an application is undeployed or upgraded to a new version, incoming requests may have encoded URIs that specify "stale" or nonexistent call or session IDs. The `stale-session-handling` element enables you to configure the action that Oracle WebLogic Communication Services takes when it encounters stale session data in a request. The following actions are possible:

- `drop`—Drops the message without logging an error. This setting is desirable for systems that frequently upgrade applications using Oracle WebLogic Communication Services's in-place upgrade feature. Using the `drop` action ensures that messages intended for older, incompatible versions of a deployed application are dropped.
- `error`—Responds with an error, so that a UAC might correct the problem. This is the default action. Messages having a `To:` tag cause a `481 Call/Transaction Does Not Exist` error, while those without the tag cause a `404 Not Found` error.
- `continue`—Ignores the stale session data and continues processing the request.

> **Note:** When it encounters stale session data, Oracle WebLogic Communication Services applies the action specified by `stale-session-handling` before considering the value of the `default-behavior` element. This means that the `default-behavior` is performed only when you have configured `stale-session-handling` to perform the `continue` action.

A.5.30 enable-contact-provisional-response

By default Oracle WebLogic Communication Services does not place a `Contact` header in non-reliable provisional (1xx) responses that have a `To` header. If you deploy applications that expect the `Contact` header to be present in such 1xx responses, set this element to true:

```
<enable-contact-provisional-response>true</enable-contact-provisional-response>
```

Note that setting this element to true does not affect 100 Trying responses.

A.5.31 app-router

The `app-router` stanza contains several elements that configure SIP Servlet v1.1 application router behavior. See Section A.5.32, "use-custom-app-router", Section A.5.33, "app-router-config-data", Section A.5.34, "custom-app-router-jar-file-name", and Section A.5.35, "default-application-name".

A.5.32 use-custom-app-router

The `use-custom-app-router` element determines whether Oracle WebLogic Communication Services uses the default, built-in Application AR (AR), or a custom AR that you specify with the `custom-app-router-jar-file-name` element. The default value, "false," configures the server to use the default AR.

A.5.33 app-router-config-data

The `app-router-config-data` element defines properties to pass to the default or custom Application Router (AR) in the `init` method. All configuration properties must conform to the Java Properties format, and each individual property must be entered on a separate, single line without line breaks or spaces. DAR properties must conform to the detailed property format described in Appendix C of `http://jcp.org/en/jsr/detail?id=289`. Example A–7 shows an example configuration.

Example A–7 Sample app-router-config-data element

```
<?xml version='1.0' encoding='UTF-8'?>
<sip-server xmlns="http://www.bea.com/ns/wlcp/wlss/300"
xmlns:sec="http://www.bea.com/ns/weblogic/90/security"
xmlns:wls="http://www.bea.com/ns/weblogic/90/security/wls"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <app-router>
    <use-custom-app-router>false</use-custom-app-router>

<app-router-config-data>INVITE:("OriginatingCallWaiting","DAR:From","ORIGINATING",
"","NO_ROUTE","0"),("CallForwarding","DAR:To","TERMINATING","","NO_ROUTE","1")
SUBSCRIBE:("CallForwarding","DAR:To","TERMINATING","","NO_
ROUTE","1")</app-router-config-data>
    <custom-app-router-jar-file-name></custom-app-router-jar-file-name>
    <default-application-name></default-application-name>
  </app-router>
</sip-server>
```

You can optionally specify AR initialization properties when starting the Oracle WebLogic Communication Services instance by including the `-Djavax.servlet.sip.ar.dar.configuration` Java option. (To specify a property file, rather than a URI, include the prefix `file:///`) If you specify the Java

startup option, the container ignores any configuration properties defined in `app-router-config-data`. You can modify the properties in at any time, but the properties are not passed to the AR until the server is restarted with the `-Djavax.servlet.sip.ar.dar.configuration` option omitted.

A.5.34 custom-app-router-jar-file-name

The `custom-app-router-jar-file-name` element specifies the filename of the custom Application Router (AR), packaged as a JAR file, to use. The custom AR implementation must reside in the *$DOMAIN_HOME*`/approuter` subdirectory.

A.5.35 default-application-name

The `default-application-name` element specifies the name of a default application that the container should call when the custom Application Router (AR) cannot find an application to process an initial request. If no default application is specified, the container returns a 500 error if the AR cannot select an application.

Note: You must first deploy an application before specifying its name as the value of **Default application name**.

B

SIP Data Tier Configuration Reference

The following sections provide a complete reference to the SIP data tier configuration file, `datatier.xml`:

B.1 Overview of datatier.xml

The `datatier.xml` configuration file identifies servers that manage the concurrent call state for SIP applications, and defines how those servers are arranged into SIP data tier *partitions*. A *partition* refers to one or more SIP data tier server instances that manage the same portion of the call state. Multiple servers in the same partition are referred to as *replicas* because they all manage a copy of the same portion of the call state.

`datatier.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where *`DOMAIN_DIR`* is the root directory of the Oracle WebLogic Communication Services domain.

B.2 Editing datatier.xml

You can edit `datatier.xml` using either the Administration Console or a text editor. Note that changes to the SIP data tier configuration cannot be applied to servers dynamically; you must restart servers in order to change SIP data tier membership or reconfigure partitions.

B.3 XML Schema

This schema file is bundled within the `wcp-sipserver.xsd` library, installed in the *`WLSS_HOME`*`/server/lib/wlss` directory.

B.4 Example datatier.xml File

Example B–1 shows the template `datatier.xml` file created using the Configuration Wizard.

Example B–1 Default datatier.xml File

```
<st:data-tier xmlns:st="http://www.bea.com/ns/wlcp/wlss/300">
 <st:partition>
   <st:name>partition-0</st:name>
   <st:server-name>replica1</st:server-name>
   <st:server-name>replica2</st:server-name>
 </st:partition>
</st:data-tier>
```

B.5 XML Element Description

`datatier.xml` contains one or more `partition` elements that define servers' membership in a SIP data tier partition. All SIP data tier clusters must have at least one `partition`. Each partition contains the XML elements described in Table B–1.

Table B–1 Nested partition Elements

Element	Description
`name`	A String value that identifies the name of the partition. Oracle recommends including the number of the partition (starting at 0) in the text of the name for administrative purposes. For example, "partition-0."
`server-name`	Specifies the name of a Oracle WebLogic Communication Services instance that manages call state in this partition. You can define up two three servers per `partition` element. Multiple servers in the same partition maintain the same call state data, and are referred to as *replicas*. Oracle recommends including the number of the server (starting with 0) and the number of the partition in the server name for administrative purposes. For example, "replica-0-0."

C

Diameter Configuration Reference

The following sections provide a complete reference to the Diameter configuration file, `diameter.xml`:

C.1 Overview of diameter.xml

The `diameter.xml` file configures attributes of a Diameter node, such as:

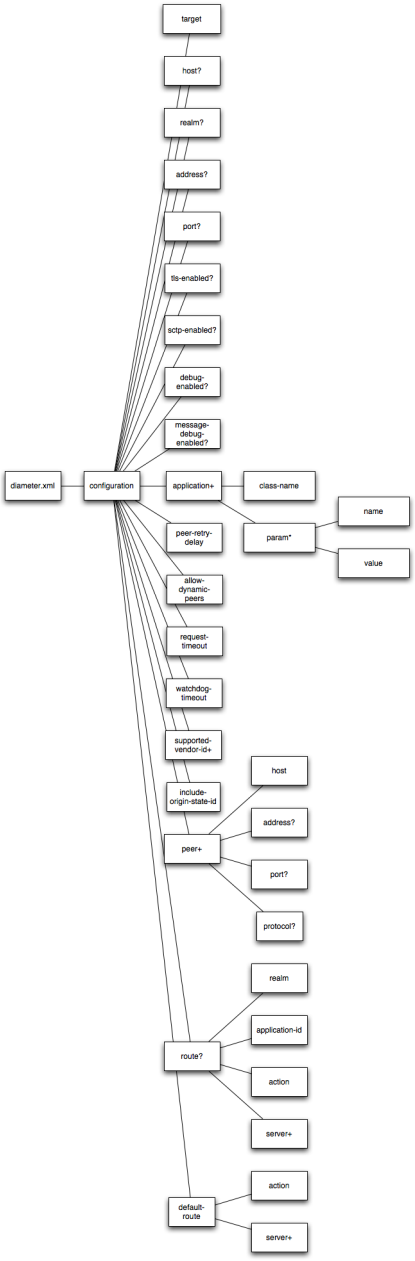
- The host identity of the Diameter node
- The Diameter applications that are deployed on the node
- Connection information for Diameter peer nodes
- Routing information and default routes for handling Diameter messages.

The Diameter protocol implementation reads the configuration file at boot time. `diameter.xml` is stored in the *`DOMAIN_DIR`*`/config/custom` subdirectory where *`DOMAIN_DIR`* is the root directory of the Oracle WebLogic Communication Services domain.

C.2 Graphical Representation

Figure C–1 shows the element hierarchy of the `diameter.xml` file.

Figure C–1 Element Hierarchy of diameter.xml



C.3 Editing diameter.xml

You should never move, modify, or delete the `diameter.xml` file during normal operations.

Oracle recommends using the Administration Console to modify `diameter.xml` indirectly, rather than editing the file by hand. Using the Administration Console ensures that the `diameter.xml` document always contains valid XML.

You may need to manually view or edit `diameter.xml` to troubleshoot problem configurations, repair corrupted files, or to roll out custom Diameter node configurations to a large number of machines when installing or upgrading Oracle

WebLogic Communication Services. When you manually edit `diameter.xml`, you must reboot Diameter nodes to apply your changes.

> **Caution:** Always use the Diameter node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide* to make changes to a running Oracle WebLogic Communication Services deployment.

C.3.1 Steps for Editing diameter.xml

If you need to modify `diameter.xml` on a production system, follow these steps:

1. Use a text editor to open the *`DOMAIN_DIR`*`/config/custom/diameter.xml` *file, where `DOMAIN_DIR`* is the root directory of the Oracle WebLogic Communication Services domain.
2. Modify the `diameter.xml` file as necessary. See Section C.6, "XML Element Description" for a full description of the XML elements.
3. Save your changes and exit the text editor.s
4. Reboot or start servers to have your changes take effect:

> **Caution:** Always use the Diameter node in the Administration Console or the WLST utility, as described in Configuring Engine Tier Container Properties in the *Configuration Guide*, to make changes to a running Oracle WebLogic Communication Services deployment.

5. Test the updated system to validate the configuration.

C.4 XML Schema

The xml schema file (`wcp-diameter.xsd`) is bundled within the `wlssdiameter.jar` library, installed in the *`WLSS_HOME`*`/server/lib/wlss` directory.

C.5 Example diameter.xml File

See Configuring Diameter Sh Client Nodes and Relay Agents in *Configuring Network Resources* for multiple listings of example `diameter.xml` configuration files.

C.6 XML Element Description

The following sections describe each XML element in `diameter.xml`.

C.6.1 configuration

The top level `configuration` element contains the entire diameter node configuration.

C.6.2 target

Specifies one or more target Oracle WebLogic Communication Services instances to which the node configuration is applied. The target servers must be defined in the `config.xml` file for your domain.

C.6.3 host

Specifies the host identity for this Diameter node. If no `host` element is specified, the identity is taken from the local server's host name. Note that the host identity may or may not match the DNS name.

> **Note:** When configuring Diameter support for multiple Sh client nodes, it is best to omit the `host` element from the `diameter.xml` file. This enables you to deploy the same Diameter Web Application to all servers in the engine tier cluster, and the host name is dynamically obtained for each server instance.

C.6.4 realm

Specifies the realm name for which this Diameter node has responsibility. You can run multiple Diameter nodes on a single host using different realms and listen port numbers. The HSS, Application Server, and relay agents must all agree on a realm name or names. The realm name for the HSS and Application Server need not match.

If you omit the `realm` element, the realm named is derived using the domain name portion of the host name, if the host name is fully-qualified (for example, host@oracle.com).

C.6.5 address

Specifies the listen address for this Diameter node, using either the DNS name or IP address. If you do not specify an address, the node uses the `host` identity as the listen address.

> **Note:** The `host` identity may or may not match the DNS name of the Diameter node. Oracle recommends configuring the `address` element with an explicit DNS name or IP address to avoid configuration errors.

C.6.6 port

Specifies the TCP or TLS listen port for this Diameter node. The default port is 3868.

C.6.7 tls-enabled

This element is used only for standalone node operation to advertise TLS capabilities.

Oracle WebLogic Communication Services ignores the `tls-enabled` element for nodes running within a server instance. Instead, TLS transport is reported as enabled if the server instance has configured a Network Channel having TLS support (a diameters channel). See Creating Network Channels for the Diameter Protocol in *Configuring Network Resources*.

C.6.8 sctp-enabled

This element is used only for standalone node operation to advertise SCTP capabilities.

Oracle WebLogic Communication Services ignores the `sctp-enabled` element for nodes running within a server instance. Instead, SCTP transport is reported as enabled if the server instance has configured a Network Channel having SCTP support (a diameter-sctp channel). See Creating Network Channels for the Diameter Protocol in *Configuring Network Resources*.

C.6.9 debug-enabled

Specifies a boolean value to enable or disable debug message output. Debug messages are disabled by default.

C.6.10 message-debug-enabled

Specifies a boolean value to enable or disable tracing of Diameter messages. This element is disabled by default.

C.6.11 application

Configures a particular Diameter application to run on the selected node. Oracle WebLogic Communication Services includes applications to support nodes that act as Diameter Sh, Ro, and Rf clients, Diameter relay agents, or Home Subscriber Servers (HSS). Note that the HSS application is a simulator that is provided only for development or testing purposes.

C.6.11.1 class-name

Specifies the application class file to load.

C.6.11.2 param*

Specifies one or more optional parameters to pass to the application class.

C.6.11.2.1 name Specifies the name of the application parameter.

C.6.11.2.2 value Specifies the value of the parameter.

C.6.12 peer-retry-delay

Specifies the number of seconds this node waits between retries to Diameter peers. The default value is 30 seconds.

C.6.13 allow-dynamic-peers

Specifies a boolean value that enables or disables dynamic peer configuration. Dynamic peer support is disabled by default. Oracle recommends enabling dynamic peers only when using the TLS transport, because no access control mechanism is available to restrict hosts from becoming peers.

C.6.14 request-timeout

Specifies the number of milliseconds to wait for an answer from a peer before timing out.

C.6.15 watchdog-timeout

Specifies the number of seconds used for the Diameter Tw watchdog timer.

C.6.16 supported-vendor-id+

Specifies one or more vendor IDs to be added to the `Supported-Version-Ids` AVP in the capabilities exchange.

C.6.17 include-origin-state

Specifies whether the node should include the origin state AVP in requests and answers.

C.6.18 peer+

Specifies connection information for an individual Diameter peer. You can choose to configure connection information for individual peer nodes, or allow any node to be dynamically added as a peer. Oracle recommends using dynamic peers only if you are using the TLS transport, because there is no way to filter or restrict hosts from becoming peers when dynamic peers are enabled.

When configuring Sh client nodes, the `peers` element should contain peer definitions for each Diameter relay agent deployed to your system. If your system does not use relay agents, you must include a peer entry for the Home Subscriber Server (HSS) in the system, as well as for all other engine tier nodes that act as Sh client nodes.

When configuring Diameter relay agent nodes, the `peers` element should contain peer entries for all Diameter client nodes that access the peer, as well as the HSS.

C.6.18.1 host

Specifies the host identity for a Diameter peer.

C.6.18.2 address

Specifies the listen address for a Diameter peer. If you do not specify an address, the host identity is used.

C.6.18.3 port

Specifies the TCP or TLS port number for this Diameter peer. The default port is 3868.

C.6.18.4 protocol

Specifies the protocol used by the peer. This element may be one of `tcp` or `sctp`.

C.6.19 route

Defines a realm-based route that this node uses when resolving messages.

When configuring Sh client nodes, you should specify a route to each Diameter relay agent node deployed in the system, as well as a `default-route` to a selected relay. If your system does not use relay agents, simply configure a single `default-route` to the HSS.

When configuring Diameter relay agent nodes, specify a single `default-route` to the HSS.

C.6.19.1 realm

The target realm used by this route.

C.6.19.2 application-id

The target application ID for the route.

C.6.19.3 action

An action type that describes the role of the Diameter node when using this route. The value of this element can be one of the following:

- none
- local
- relay
- proxy
- redirect

C.6.19.4 server+

Specifies one or more target servers for this route. Note that any server specified in the `server` element must also be defined as a `peer` to this Diameter node, or dynamic peer support must be enabled.

C.6.20 default-route

Defines a default route to use when a request cannot be matched to a configured route.

C.6.20.1 action

Specifies the default routing action for the Diameter node. See Section C.6.19.3, "action".

C.6.20.2 server+

Specifies one or more target servers for the default route. Any server you include in this element must also be defined as a `peer` to this Diameter node, or dynamic peer support must be enabled.

D

Startup Command Options

Table D–1 provides a reference to the startup configuration options available to Oracle WebLogic Communication Services and other Oracle WebLogic Communication Services utilities.

Table D–1 Startup Command Options

Application	Startup Option Link
SIP Servlet Application Router	-Djavax.servlet.sip.dar.configuration
	-Djavax.servlet.sip.ar.spi.SipApplicationRouterProvider
Oracle WebLogic Communication Services	-Dwlss.udp.listen.on.ephemeral
	-Dwlss.udp.lb.masquerade
	-Dweblogic.management.discover
	-Dweblogic.RootDirectory
Oracle WebLogic SIP Server	-Dwlss.udp.listen.on.ephemeral
	-Dwlss.udp.lb.masquerade
	-Dweblogic.management.discover
	-Dweblogic.RootDirectory
	-DWLSS.SNMPAgentPort
Installer	-Djava.io.tmpdir
WlssEchoServer	-Dwlss.ha.echoserver.port
	-Dwlss.ha.echoserver.logfile
	-Dreplica.host.monitor.enabled
	-Dwlss.ha.heartbeat.interval
	-Dwlss.ha.heartbeat.count
	-Dwlss.ha.heartbeat.SoTimeout

For more information about these, and other options, see *Oracle Fusion Middleware Command Reference for Oracle WebLogic Server*.

E

Supported Platforms, Protocols, RFCs and Standards

The following sections describe how Oracle WebLogic Communication Services complies with various specifications and RFCs:

- Section E.1, "Supported Configurations"
- Section E.2, "Supported SIP Clients"
- Section E.3, "Supported Load Balancer"
- Section E.4, "Supported Databases"
- Section E.5, "Overview of Oracle WebLogic Communication Services Standards Alignment"
- Section E.6, "Java Sun Recommendation (JSR) Standards Compliance"
- Section E.7, "IETF RFC Compliance"
- Section E.8, "3GPP R6 Specification Conformance"

E.1 Supported Configurations

Oracle WebLogic Communication Services is supported for production use on the operating system, hardware, and JVM combinations described shown in http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

> **Note:** Production deployment is supported only for Linux and UNIX platforms. Windows platforms are supported only for development purposes.

Oracle WebLogic Communication Services has similar requirements to Oracle WebLogic Server 10*g* Release 3. The following items are required in addition to the basic WebLogic Server requirements:

- Gigabit Ethernet connections are required between engine and SIP data tier servers for most production deployments.
- Dual NICs are required to provide failover capabilities in a production environment.
- Additional RAM is required to support the throughput requirements of most production installations.

- A compatible Load Balancer is required for production installations.

E.2 Supported SIP Clients

Oracle WebLogic Communication Services and the included sample applications support the following SIP softphone:

- Oracle Communicator (default client)

E.3 Supported Load Balancer

Oracle WebLogic Communication Services has been tested to run with F5 Networks, Inc. BIG-IP load balancer, versions 9.0.5 through 9.2.3 Build 34.3.

See the Oracle Technology Network (http://www.oracle.com/global/index.html) for information about support for other load balancer solutions.

E.4 Supported Databases

Oracle WebLogic Communication Services has been tested to run with the following databases for storing long-lived session data and for replicating call state information across regional sites (geographic persistence):

- Oracle Database 10*g* (10.2.0.3 or higher)
- Oracle Database 11*g* (11.1.0.6 or higher)

E.5 Overview of Oracle WebLogic Communication Services Standards Alignment

Oracle WebLogic Communication Services is developed with special attention to Internet Engineering Task Force and 3rd Generation Partnership Project specifications. Feature development is prioritized according to general market trends, both observed and predicted. In cases where certain specifications are obsolete or where Internet drafts are formalized as 'Request For Comments' standards, Oracle WebLogic Communication Services places priority on compliance with those specifications. In cases where specifications are part of a larger release plan, as with the 3GPP, Oracle prioritizes compliance with the latest ratified release (in this case, Release 6). This should not be presumed to mean that the product is not compliant with subsequent versions of component specifications, although this document does not summarize compliance with those specifications.

E.6 Java Sun Recommendation (JSR) Standards Compliance

Oracle WebLogic Communication Services is compliant with Java EE version 5.0 and the corresponding Java EE component specifications.

Oracle WebLogic Communication Services is further enhanced by the addition of a SIP Servlet container defined by JSR 289: "SIP Servlet API."

Oracle WebLogic Communication Services has executed all related Test Compatibility Kits (TCKs) and has met the formal requirements established by Sun Microsystems for formal public statements of compliance.

E.7 IETF RFC Compliance

The following table lists the Oracle WebLogic Communication Services level of compliance to common Internet Engineering Task Force (IETF) Requests for Comment (RFCs) and Internet drafts. The level of compliance is defined as follows:

- **Yes**—Indicates that Oracle WebLogic Communication Services directly supports the feature or specification.
- **Yes (Platform)**—Indicates Oracle WebLogic Communication Services can host applications or components that implement the RFC. However, the RFC or feature has no impact on the transaction layer of the protocol or on the behavior of the SIP Servlet container.

Table E–1 Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
761	DoD Standard Transmission Control Protocol	Yes	See http://www.ietf.org/rfc/rfc761.txt
768	User Datagram Protocol	Yes	See http://www.ietf.org/rfc/rfc768.txt
1157	A Simple Network Management Protocol (SNMP)	Yes	Oracle WebLogic Communication Services supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1157.txt
1847	Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted	Yes (Platform)	Oracle WebLogic Communication Services supports applications that consume or generate signed or encrypted multipart MIME objects. See http://www.ietf.org/rfc/rfc1847.txt
1901	Introduction to Community-based SNMPv2	Yes	Oracle WebLogic Communication Services supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1901.txt
1905	Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes	Oracle WebLogic Communication Services supports SNMP V2c traps. See http://www.ietf.org/rfc/rfc1905.txt
1906	Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes	Oracle WebLogic Communication Services supports SNMP over both TCP and UDP. See http://www.ietf.org/rfc/rfc1906.txt
1907	Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)	Yes (Platform)	See http://www.ietf.org/rfc/rfc1907.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
2183	Communicating Presentation Information in Internet Messages: The Content-Dispositio n Header Field	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc2183.txt
2246	The TLS Protocol Version 1.0	Yes	Oracle WebLogic Communication Services supports TLS. See http://www.ietf.org/rfc/rfc2246.txt
2327	SDP: Session Description Protocol	Yes	Oracle WebLogic Communication Services supports applications that consume or generate SDP. See http://www.ietf.org/rfc/rfc2327.txt
2460	Internet Protocol, Version 6 (IPv6) Specification	Yes	See http://www.ietf.org/rfc/rfc2460.txt
2543	SIP: Session Initiation Protocol (v1)	Yes	Oracle WebLogic Communication Services supports backward compatibility as described in this specification. See http://www.ietf.org/rfc/rfc2543.txt
2616	Hypertext Transfer Protocol -- HTTP 1.1	Yes	See http://www.ietf.org/rfc/rfc2616.txt
2617	HTTP Authentication: Basic and Digest Access Authentication	Yes	See http://www.ietf.org/rfc/rfc2617.txt
2782	A DNS RR for specifying the location of services (DNS SRV)	Yes	See http://www.ietf.org/rfc/rfc2782.txt
2806	URLs for Telephone Calls	Yes	See http://www.ietf.org/rfc/rfc2806.txt
2848	The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services	Yes (Platform)	Note that implementing PINT services implies a pre-IMS architecture. Although Oracle favors the 3GPP/TISPAN architecture and approach to class 4/5 Service Emulation and does not advocate PINT, it is possible to implement PINT service elements using Oracle WebLogic Communication Services. See http://www.ietf.org/rfc/rfc2848.txt
2916	E.164 number and DNS	Yes	See http://www.ietf.org/rfc/rfc2916.txt
2960	Stream Control Transmission Protocol	Yes	SCTP supported only for Diameter traffic. See http://www.ietf.org/rfc/rfc2960.txt
2976	The SIP INFO Method	Yes	See http://www.ietf.org/rfc/rfc2976.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3204	MIME media types for ISUP and QSIG Objects	Yes (Platform)	Oracle WebLogic Communication Services does not directly consume or generate ISUP and QSIG objects, but it supports applications that consume or generate these objects. See http://www.ietf.org/rfc/rfc3204.txt
3261	SIP: Session Initiation Protocol	Yes	See http://www.ietf.org/rfc/rfc3261.txt
3262	Reliability of Provisional Responses in the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/rfc/rfc3262.txt
3263	Session Initiation Protocol (SIP): Locating SIP Servers	Yes	See http://www.ietf.org/rfc/rfc3263.txt
3264	An Offer/Answer Model with Session Description Protocol (SDP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3264.txt
3265	Session Initiation Protocol (SIP)-Specific Event Notification	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3265.txt
3266	Support for IPv6 in Session Description Protocol (SDP)	Yes	See http://www.ietf.org/rfc/rfc3266.txt
3268	Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)	Yes (Platform)	Oracle WebLogic Communication Services supports cryptographic services, but specific algorithms that are used are subject to local availability and export control. See http://www.ietf.org/rfc/rfc3268.txt
3311	The Session Initiation Protocol (SIP) UPDATE Method	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3311.txt
3312	Integration of Resource Management and Session Initiation Protocol (SIP).	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3312.txt
3313	Private Session Initiation Protocol (SIP) Extensions for Media Authorization	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3313.txt
3323	A Privacy Mechanism for the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3323.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3325	Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks	Yes	See http://www.ietf.org/rfc/rfc3325.txt
3326	The Reason Header Field for the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3326.txt
3327	Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts.	Yes (Platform)	See http://www.ietf.org/rfc/rfc3327.txt
3351	User Requirements for the Session Initiation Protocol (SIP) in Support of Deaf, Hard of Hearing and Speech-impaired Individuals	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3351.txt
3372	Session Initiation Protocol for Telephones (SIP-T): Context and Architectures	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3372.txt
3388	Grouping of Media Lines in the Session Description Protocol (SDP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3388.txt
3420	Internet Media Type message/sipfrag		See http://www.ietf.org/rfc/rfc3420.txt
3428	Session Initiation Protocol (SIP) Extension for Instant Messaging	Yes	See http://www.ietf.org/rfc/rfc3428.txt
3455	Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3455.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3489	STUN-SimpleTraversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)	Yes	See http://www.ietf.org/rfc/rfc3489.txt
3515	The Session Initiation Protocol (SIP) Refer Method.	Yes	See http://www.ietf.org/rfc/rfc3515.txt
3524	Mapping of Media Streams to Resource Reservation Flows	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3524.txt
3556	Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3556.txt
3578	Mapping of Integrated Services Digital Network (ISDN) User Part (ISUP) Overlap Signalling to the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification, but it does not provide an ISUP interface. See http://www.ietf.org/rfc/rfc3578.txt
3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing	Yes	See http://www.ietf.org/rfc/rfc3581.txt
3589	Diameter Command Codes for Third Generation Partnership Project (3GPP) Release 5	Yes	See http://www.ietf.org/rfc/rfc3589.txt
3588	Diameter Base Protocol	Yes	See http://www.ietf.org/rfc/rfc3588.txt
3605	Real Time Control Protocol (RTCP) attribute in Session Description Protocol ((SDP)		See http://www.ietf.org/rfc/rfc3605.txt
3608	Session Initiation Protocol (SIP) Extension Header Field for Service Route Discovery During Registration.	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification, but it does not provide a means of storing the ServiceRoute established during registration. This functionality can be implemented as part of the application. See http://www.ietf.org/rfc/rfc3608.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3665	Session Initiation Protocol (SIP) Basic Call Flow Examples.	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3665.txt
3666	Session Initiation Protocol (SIP) Public Switched Telephone Network (PSTN) Call Flows	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3666.txt
3680	A Session Initiation Protocol (SIP) Event Package for Registrations	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3680.txt
3689	General Requirements for Emergency Telecommunication Service (ETS)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3689.txt
3690	IP Telephony Requirements for Emergency Telecommunication Service (ETS)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3690.txt
3702	Authentication, Authorization, and Accounting Requirements for the Session Initiation Protocol (SIP)	Yes	Oracle WebLogic Communication Services version supports JDBC and LDAP. See http://www.ietf.org/rfc/rfc3702.txt
3725	Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3725.txt
3761	The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)	Yes	See http://www.ietf.org/rfc/rfc3761.txt
3764	Enumservice Registration for Session Initiation Protocol (SIP) Addresses-of-Record	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3764.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3824	Using E.164 numbers with the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/rfc/rfc3824.txt
3840	Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3840.txt
3841	Caller Preferences for the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3841.txt
3853	S/MIME Advanced Encryption Standard (AES) Requirement for the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3853.txt
3856	Presence Event Package for the Session Initiation Protocol (SIP)	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3856.txt
3857	Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3857.txt
3858	Extensible Markup Language (XML) Based Format for Watcher Information	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3858.txt
3863	Presence Information Data Format (PIDF)	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3863.txt
3891	The Session Initiation Protocol (SIP) 'Replaces' Header	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3891.txt
3892	The Session Initiation Protocol (SIP) Referred-By Mechanism	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3892.txt
3893	Session Initiation Protocol (SIP) Authenticated Identity Body (AIB) Format	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3893.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
3903	Session Initiation Protocol (SIP) Extension for Event State Publication	Yes	See http://www.ietf.org/rfc/rfc3903.txt
3911	The Session Initiation Protocol (SIP) "Join" Header	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3911.txt
3959	The Early Session Disposition Type for the Session Initiation Protocol (SIP)		See http://www.ietf.org/rfc/rfc3959.txt
3960	Early Media and Ringing Tone Generation in the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc3960.txt
3966	The tel URI for Telephone Numbers	Yes	See http://www.ietf.org/rfc/rfc3966.txt
4028	Session Timers in the Session Initiation Protocol (SIP)	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4028.txt
4032	Update to the Session Initiation Protocol (SIP) Preconditions Framework	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4032.txt
4244	An Extension to the Session Initiation Protocol (SIP) for Request History Information	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4244.txt
4320	Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction		See http://www.ietf.org/rfc/rfc4320.txt
4321	Problems Identified Associated with the Session Initiation Protocol's (SIP) Non_INVITE Transaction		See http://www.ietf.org/rfc/rfc4321.txt
4474	Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/rfc/rfc4474.txt.

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
4479	Data Model for Presence	Yes	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4479.txt.
4480	RPID: Rich Presence Extensions to the Presence Information Data Format	Implicitly Supported	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4480.txt.
4481	Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Status Information for Past and Future Time Intervals	Implicitly Supported	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4481.txt.
4483	A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See http://www.ietf.org/rfc/rfc4483.txt.
4825	The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)	Partially Supported	No support for partial document manipulation (XPath)
4827	Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents	Yes	Replaces draft-ietf-simple-xcap-pidf-manipulation-usage-02
draft-levy-sip-diversion-08	Diversion Indication in SIP	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See https://datatracker.ietf.org/public/idindex.cgi?command=id_detail&id=6002
draft-donovan-mmusic-183-00	SIP 183 Session Progress Message Draft	Yes (Platform)	Oracle WebLogic Communication Services supports applications that conform to this specification. See https://datatracker.ietf.org/public/idindex.cgi?command=id_detail&id=4308
draft-ietf-sip-gruu-15	Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)	Yes	See http://www.ietf.org/internet-drafts/draft-ietf-sip-gruu-15.txt

Table E–1 (Continued)Oracle WebLogic Communication Services IETF Compliance

RFC or Specification Number	Title	Compliant?	Additional Information
Presence Authorization Rules	draft-ietf-simple-presence-rules-04	Yes	http://tools.ietf.org/html/draft-ietf-simple-presence-rules-10
OMA extensions to the presence data model	OMA-TS-Presence_SIMPLE-V1_0-20051122-C	Yes	
OMA extensions to geopriv common policy	OMA-TS-XDM_Core-V1_0-20051122-C	Yes	
OMA extensions to presence rules	OMA-TS-Presence_SIMPLE_XDM-V1_0-20051122-C	Yes	

E.8 3GPP R6 Specification Conformance

Table E–2 summarizes the ability of Oracle WebLogic Communication Services to support implementation of the enablers or application functions identified by each applicable 3GPP Release 6 specification.

Other than the exceptions noted, Oracle WebLogic Communication Services does not impose any restrictions on implementing applications or functions that are compliant with those associated with the Application Server entity described in the specification. In some cases, applications must implement support for SIP methods or headers. The default behavior of the Oracle WebLogic Communication Services Sip Servlet Container is to pass unrecognized headers, request methods and payloads to SIP Servlets using normal SIP Servlet API procedures.

Table E–2 3GPP R6 Specification Conformance

Specification	Comments
3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2 (Release 6)"	▪ No comments.
3GPP TS 24.229: "IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 (Release 6)"	▪ Oracle WebLogic Communication Services does not enforce the requirement that only one p-charging-function-address header per SIP request as described in sub-section 5.7.1.2. Oracle WebLogic Communication Services does enforce uniqueness. ▪ Oracle WebLogic Communication Services does not provide privacy support as described in sub-section 5.7.3.

Table E–2 (Continued)3GPP R6 Specification Conformance

Specification	Comments
3GPP TS 23.141: "Presence Service; Architecture and Functional description (Release 6)"	■ Oracle WebLogic Communication Services does not natively support the Ph (MAP), Pl (LIF-MLP), Px (DIAMETER Cx/Dx), Pg (phase 4, 3GPP Release 5),Pc (CAMEL phase 4, 3GPP Release 5) or Pr, Pk and Pp (RADIUS) reference points. ■ Oracle WebLogic Communication Services does not support IPv6 as required for the Presence User Agent (Peu) reference point as required in sub-section 4.3.1.
3GPP TS 23.218: "IP Multimedia (IM) session handling; IM call model; Stage 2 (Release 6)"	■ No comments.
3GPP TS 24.247 "Messaging using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3 (Release 6)"	■ Oracle WebLogic Communication Services does not provide support for the Message Session Relay Protocol (MSRP), although it is presumed that an MSRP relay will typically be implemented as a Media Resource Function in the IMS architecture.
3GPP TS 24.841: "Presence service based on Session Initiation Protocol (SIP); Functional models, information flows and protocol details (Release 6)"	■ Oracle WebLogic Communication Services does not provide support for IETF RFC 3310: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)".
3GPP TS 24.109: "Bootstrapping interface (Ub) and Network application function interface (Ua); Protocol details (Release 6)"	■ Oracle WebLogic Communication Services does not provide support for IETF RFC 3310: "Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA)". ■ Oracle WebLogic Communication Services supports the 'X-3GPP-Asserted-Identity extension-header' for use in applying access control and authorization constraints within the integrated security framework.
3GPP TS 29.328: "IP Multimedia Subsystem (IMS) Sh interface; Signalling flows and message contents"	■ No comments.
3GPP TS 29.329: "Sh interface based on the Diameter protocol; Protocol details"	■ No comments.
3GPP TS 32.299: "Telecommunication management; Charging management; Diameter charging applications"	■ No comments.
3GPP TS 33.222: "Generic Authentication Architecture (GAA); Access to network application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (Release 6)"	■ Oracle WebLogic Communication Services supports the Application Server role in the GAA.

F

Using Oracle WebLogic Communication Services Export/Import

The following sections describe how to use Oracle WebLogic Communication Services Export and Import functions:

- Section F.1, "Export"
- Section F.2, "Import"

> **Note:** All examples in this document use Linux commands; adapt commands as appropriate for other operating systems.

F.1 Export

This section details the steps you must take in order to export your OWLCS data.

F.1.1 Export the Database Data from the Current Environment

> **Note:** The first four steps in this procedure are performed on your RDBMS machine.

Export data by using one of the Oracle database export utilities. Before executing the export commands, on the existing database machine, create a directory on the file system where the exported data dump files and log files will be stored.

```
$ mkdir ~/owlcs_data_pump_dir
```

1. Change directory to the `$ORACLE_HOME/bin` directory of your database installation and execute the following commands. These commands can be executed as the *sys* user or the *system* user.
2. Create `DATA_PUMP_DIR` in your database using the following commands:
   - `$ ./sqlplus sys@<db_service> as sysdba`
   - Enter password for sys user at the prompt.
   - At the SQL prompt, execute the following commands:

```
SQL> create or replace directory DATA_PUMP_DIR as '<full_path_to_the_data_
pump_dir_created_above>';
SQL> commit;
SQL> quit
```

3. Export data in the subscriber data services schema.

   - `$ ./expdp "'sys@<db_service> as sysdba'" SCHEMAS=<current_prefix>_orasdpsds DIRECTORY=DATA_PUMP_DIR DUMPFILE=<current_prefix>_orasdpsds.dmp LOGFILE=<current_prefix>_orasdpsds.log`
   - Enter password for sys user at the prompt.

   Note: The value of the DIRECTORY parameter in the command line above should be specified as DATA_PUMP_DIR and not as the actual directory on the file system. When the command is done, verify that two files, one corresponding to DUMPFILE and the other corresponding to LOGFILE are created in your data pump directory.

4. Export data in the XDMS schema.

   - `$ ./expdp "'sys@<db_service> as sysdba'" SCHEMAS=<current_prefix>_orasdpsds DIRECTORY=DATA_PUMP_DIR DUMPFILE=<current_prefix>_orasdpsds.dmp LOGFILE=<current_prefix>_orasdpsds.log`
   - Enter password for sys user at the prompt.

   Note: The value of the DIRECTORY parameter in the command line above should be specified as DATA_PUMP_DIR and not as the actual directory on the file system. When the command is done, verify that two files, one corresponding to DUMPFILE and the other corresponding to LOGFILE are created in your data pump directory.

   Exporting of the Location Service schema is not required, since location data is recreated when clients sign in to the server.

5. Complete OWSM Policy Migration. Start WLST by running the following command:

   ```
   $ORACLE_HOME/common/bin/wlst.sh.
   ```

6. Connect to the local WLS instance by running the following command:

   ```
   wls:/offline> connect('weblogic','weblogic','127.0.0.1:7001')
   ```

   In this example, 'weblogic'/'weblogic' are sample WLS admin username/password. Replace them with the real values in your environment. The port may change if you have another instance of WLS running (this is the WLS AdminServer port).

7. Run the following WLST commands to export the policies and assertion templates. Replace "wlcs_server1" with your OWLCS instance name.

   ```
   wls:/base_domain/serverConfig>
   exportMetadata(application='wsm-pm',server='wlcs_
   server1',docs='/assertiontemplates/**',toLocation='/tmp/owsmexport/')
   wls:/base_domain/serverConfig>
   exportMetadata(application='wsm-pm',server='wlcs_
   server1',docs='/policies/**',toLocation='/tmp/owsmexport/')
   ```

8. Exit the WLST command line tool by running the following command:

```
wls:/base_domain/serverConfig> exit()
```

9. (Performed on the machine in which the OWLCS instance is installed [your middleware machine]) Next, export the *Credential Store*. OWSM stores client policy username and password credentials and keystore passwords in the credential store. Copy `<domain>/config/fmwconfig/cwallet.sso` from current machine to the new machine. If this is already performed as part of OPSS migration, then you can omit this step.

10. Export UMS-related details to the new environment (if UMS is installed in your environment). Export UMS-related details to your new environment (if UMS is installed in your environment). If the User Messaging Service is used in the current and new environments, perform the following step:

    - Start WLST by running the following command:

```
$ORACLE_HOME/common/bin/wlst.sh
```

> **Note:** This is the ORACLE_HOME on the middleware instance. By default that is `<middleware_home>/as11gr1wlcs1 directory`, where `<middleware_home>` is the directory where OWLCS is installed.

11. Run the following WLST commands to download the user messaging preferences from the backend database to the specified xml file:

```
wls:/offline> manageUserMessagingPrefs(operation='download',
filename='/tmp/userprefs-dump.xml', url='t3://localhost:8001',
username='weblogic', password='weblogic')
```

> **Note:** In the above sample 'weblogic'/'weblogic' are sample WLS admin username/password combinations. Replace them with the real values in your environment. *8001* is the Managed Server Port where UMS is running. Replace it accordingly with the appropriate value.

12. Exit the WLST command line tool by running the following command:

```
wls:/offline> exit()
```

F.2 Import

This section details the steps you must take in order to import your OWLCS data.

1. Import Database data into the Production Environment. This is achieved by using one of the Oracle database import utilities. Before executing the import commands, on the new database machine, create a directory on the file system where the data dump files to be imported and log files will be stored:

```
$ mkdir ~/owlcs_data_pump_dir
```

2. Copy (ftp or remote copy) all dmp files from the data pump directory on your old database machine to the above directory.

3. Change directory to the `$ORACLE_HOME/bin` of your production database installation and execute the following commands. These commands can be executed as the *sys* user or the *system* user.

a. Create DATA_PUMP_DIR in your production database with the following commands:

```
$ ./sqlplus sys@<db_service> as sysdba
```

Enter password for sys user at the prompt

At the SQL prompt, execute the following command:

```
SQL> create or replace directory DATA_PUMP_DIR as '<full_path_to_the_data_
pump_dir_created_above>'
SQL> commit;
SQL> quit
```

b. Drop all sequences in the *<production_prefix>_orasdpsds* schema with the following commands:

```
# $ ./sqlplus sys@<db_service> as sysdba
```

Enter password for sys user at the prompt

At the SQL prompt, execute the following commands:

```
SQL> alter session set current_schema=<production_prefix>_orasdpsds;
SQL> drop sequence ACCOUNT_SEQ;
SQL> drop sequence CREDENTIALS_SEQ;
SQL> drop sequence PRIVATE_IDENTITY_SEQ;
SQL> drop sequence PUBLIC_IDENTITY_SEQ;
SQL> drop sequence REALM_SEQ;
SQL> drop sequence ROLE_SEQ;
SQL> commit;
SQL> quit
```

c. Import data into the subscriber data services schema

```
$ ./impdp "'sys@<db_service> as sysdba'" REMAP_SCHEMA=<test_prefix>_
orasdpsds:<production_prefix>_orasdpsds TABLE_EXISTS_ACTION=REPLACE
DIRECTORY=DATA_PUMP_DIR DUMPFILE=<test_prefix>_orasdpsds.dmp
LOGFILE=<production_prefix>_orasdpsds.log
```

Enter password for sys user at the prompt.

Note that the value of the DIRECTORY parameter in the command line above should be specified as DATA_PUMP_DIR and not as the actual directory on the file system. Ignore error that user already exists.

d. Export data into the XDMS schema

```
$ ./impdp "'sys@<db_service> as sysdba'" REMAP_SCHEMA=<test_prefix>_
orasdpxdms:<production_prefix>_orasdpxdms TABLE_EXISTS_ACTION=REPLACE
DIRECTORY=DATA_PUMP_DIR DUMPFILE=<test_prefix>_orasdpxdms.dmp
LOGFILE=<production_prefix>_orasdpxdms.log
```

Enter password for sys user at the prompt.

Note that the value of the DIRECTORY parameter in the command line above should be specified as DATA_PUMP_DIR and not as the actual directory on the file system. Ignore error that user already exists.

Importing of the Location Service schema is not required, since location data is recreated when clients sign in to the server.

4. Copy the `/tmp/owsmexport` directory to the new machine.
5. Start WLST by running the following command:

```
$ORACLE_HOME/common/bin/wlst.sh
```

6. Connect to the local WLS instance by running the following command:

```
wls:/offline> connect('weblogic','weblogic','127.0.0.1:7001')
```

> **Note:** In the above sample 'weblogic'/'weblogic' are sample WLS admin username/password. Replace them with the real values in your environment. The port may change if you have another instance of WLS running (this is the WLS AdminServer port).

7. Run the following WLST commands to replace the policies and assertion templates. Replace *wlcs_server1* with your OWLCS instance name:

```
wls:/base_domain/serverConfig>
deleteMetadata(application='wsm-pm',server='wlcs_
server1',docs='/assertiontemplates/**')
wls:/base_domain/serverConfig>
deleteMetadata(application='wsm-pm',server='wlcs_server1',docs='/policies/**')
wls:/base_domain/serverConfig>
importMetadata(application='wsm-pm',server='wlcs_
server1',docs='/assertiontemplates/**',fromLocation='/tmp/owsmexport/')
wls:/base_domain/serverConfig>
importMetadata(application='wsm-pm',server='wlcs_
server1',docs='/policies/**',fromLocation='/tmp/owsmexport/')
```

8. Exit the WLST command line tool by running the following command:

```
wls:/base_domain/serverConfig> exit()
```

> **Note:** Regarding importing OWSM Keystore:
>
> - Private keys will differ between current and new environments. So, they do not need to be migrated.
> - Public keys, intermediate certs, and root certs may be migrated from current to new environments. Use java keytool export and import commands to move them. After doing so, review if these certs are applicable in the new environment based on the clients invoking the services in the new environment.
> - Review if new public keys of client certs, intermediate CA certs or root CA certs must be added to the new keystore based on the clients invoking the services in the new environment.
>
> For more details, see *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

9. Go to the new machine.
10. Start WLST by running the following command:

```
$ORACLE_HOME/common/bin/wlst.sh
```

11. Run the following WLST command to upload the User Messaging Preferences from file to the backend database:

```
wls:/offline> manageUserMessagingPrefs(operation='upload',
filename='/tmp/userprefs-dump.xml', url='t3://localhost:8001',
username='weblogic', password='weblogic')
```

Note: In the above sample 'weblogic'/'weblogic' are sample WLS admin username/password. Replace them with the real values in your environment. *8001* is the Managed Server Port where UMS is running. Replace it accordingly with the appropriate value.

12. Observe the message displayed for successful upload. Exit the WLST command line tool by running the following command:

```
wls:/offline> exit()
```

Note: For different options on performing download or upload, execute help ('manageUserMessagingPrefs') at the wls:/offline> prompt. User devices provisioned in the LDAP store are dynamic; the assumption is that both the current and new environments will point to the same LDAP store or will be re-configured to use the same set of information.

G

Deploying a Scalable Presence Deployment

This section describes the recommended and supported deployment topology for a large scale Presence Solution requiring Presence, XDMS, and User Dispatcher. It illustrates the typical flows from a multi-node perspective. Topics include:

G.1 Presence Cluster

A Presence Cluster is defined as a set of Presence Nodes connected after one or more Load Balancers. The Presence Cluster is responsible for processing incoming subscribe and publish requests made towards the presence event-package and for sending out notify's whenever appropriate. The Presence Cluster will also accept and process subscribe requests for the presence.winfo event-package.

The Presence Cluster will interact with the XDM Cluster in order to obtain information needed to complete its responsibilities. The information queried of the XDM Cluster is user's presence-rules and pidf-manipulation documents.

The Presence Cluster is layered into the following three distinct tiers:

- The load-balancing layer, responsible for dispatching incoming traffic to the User Dispatchers. The load balancers are stateless and are not required to understand SIP as a protocol.
- The user-dispatching layer, responsible for dispatching traffic based on user information. A user is assigned to a particular Presence Server instance and all traffic destined to that user will be dispatched to the same Presence Server instance. Even though each User Dispatcher is stateless and does not share state with the other User Dispatchers, they still need to have the same view of the Presence Server tier.
- The bottom layer is where the Presence Server instances reside. Each instance is separated from the others and does not share any state with any other instances. The purpose of the Presence Server tier is to serve incoming SUBSCRIBE and PUBLISH requests destined to the presence event-package as well as servicing subscriptions to the presence.winfo event-package.

The Presence Cluster consists of the following physical nodes:

- The Load Balancer, such as an BigIP from F5 Networks.
- The Presence Node, which consists of the following components:
  - User Dispatcher
  - Presence Server

G.2 XDM Cluster

The XDM cluster is defined as a set of XDM Nodes connected after one or more Load Balancers. The XDM cluster processes all XDM related traffic, that is, SIP subscribe traffic towards the ua-profile event-package and XCAP traffic. As such, it deals with everything that has to do with manipulating XML documents. The XDM Cluster uses a database for actual storage of the XML documents but note that the database, and potentially its cluster, is not part of the XDM Cluster.

The XDM cluster consists of the following layers:

- The load-balancing tier, responsible for dispatching both SIP and XCAP traffic to the next layer. For XCAP traffic the next tier is the Aggregation Proxy but for SIP, the traffic goes directly to the User Dispatcher layer.
- Aggregation Proxy layer – authenticates incoming traffic and upon successful authentication it forwards the requests to the User Dispatcher layer. All XCAP traffic from external networks (such as from outside the XDM cluster) goes through the Aggregation Proxy layer. Internal traffic, however, will not go through the Aggregation Proxy but rather directly to the User Dispatchers.
- User Dispatcher layer – from a SIP perspective it carries out the exact same duties and functions as in the Presence Cluster (it is the same kind of traffic after all). The main difference in the XDM Cluster compared to the presence one is that in the XDM Cluster the User Dispatchers will also have to handle XCAP traffic. However, the XCAP traffic is treated in the exact same way as SIP and the purpose of the User Dispatcher for XCAP traffic is the same as for SIP: to extract user information based on the request and then dispatch it to the correct XDMS instance.
- The XDM Server layer has the same function as the Presence Servers in the Presence Cluster. The XDMS instances serve incoming SUBSCRIBE requests for the event-package ua-profile and will send out NOTIFY messages to registered subscribers as appropriate. Note that the XDMS does not accept PUBLISH requests and updating the state of the Resources (which are XML documents) is achieved through XCAP operations. An XDM Client can manipulate the documents managed by an XDMS by issuing appropriate XCAP operations. A successful XCAP operation may alter the content of a document, causing the XDMS to send out NOTIFY messages to all subscribers of that document informing them about the change. Whenever the XDMS needs to get an XML document it queries the next layer, the database layer.
- The Database tier physically stores the XML documents managed by the XDMS. This tier guarantees high-availability and scalability so that if one of the nodes in the database layer fails, documents that resided on that node will still be accessible to the XDMS without any loss of data or service.

The XDM Cluster consists of the following physical nodes:

- The Load Balancer, such as an Big IP from F5 Networks.
- The XDM Node, which consists of the following components:

  - Aggregation Proxy
  - User Dispatcher
  - The XDM Server (XDMS)
- The database.

G.3 Presence Node

The Presence Node is the main component in the Presence Cluster and is responsible for dispatching the incoming traffic to the correct Presence Server instance and servicing users with presence information. The User Dispatcher servers the same purpose both in a single node deployment and in a multi-node deployment, which is to dispatch incoming traffic to a particular Presence Server instance; whether or not the target Presence Server instance is on the same physical node as the User Dispatcher is of no significance to the User Dispatcher.

A Presence Node will always have a User Dispatcher deployed that serves as the main entrance into the node itself. Typically, the User Dispatchers listen to port 5060 and the Presence Servers on that node listen on other ports. In this way, a single node will appear as one Presence Server to clients but is in fact multiple instances running behind the User Dispatcher. Each of the components deployed on the Presence Node is executing in their own separate Java Virtual Machine. That is, the User Dispatcher and the Presence Server instances execute in their own OWLCS and SIP containers. The reason for this is to be able to utilize all the available memory on that machine.

G.4 XDM Node

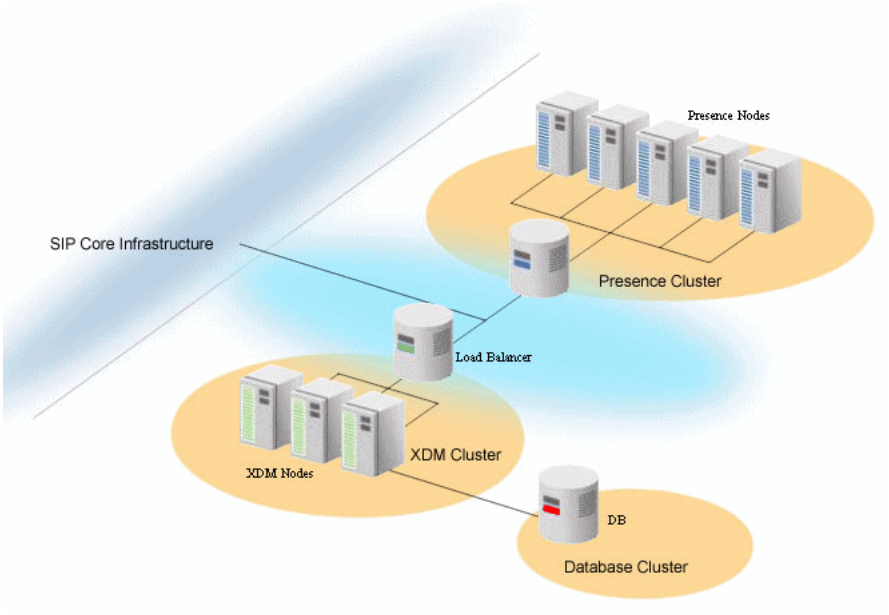
The XDM Node always has an Aggregation Proxy deployed that typically listens on port 80 for XCAP traffic. The Aggregation Proxy authenticates incoming traffic and upon successful authentication forwards the request to the User Dispatcher. As with the Presence Node, the XDM Node will also have a User Dispatcher deployed (usually on port 5060) and for SIP traffic there is no difference between the XDM and Presence Nodes. The difference between the two types of nodes is that the User Dispatcher will also dispatch XCAP traffic. As it does with SIP, it extracts the user id out of the request and, based on that, maps the request to a particular XDMS instance to which it forwards the request.

There will be a number of XDMS instances deployed to which the User Dispatcher dispatches both SIP and XCAP traffic. Just as in the case of the Presence Server instances on the Presence Node, each XDMS instance is not aware of the others and executes in isolation.

G.5 Complete Presence and XDM Cluster

Figure G–1 shows a complete Presence and XDM cluster with all necessary components. This figure also illustrates that the two clusters, Presence and XDM, are treated as two separate clusters and the way into those two networks for initial traffic is always through their respective Load Balancers. Even the Presence Servers will actually go through the Load Balancer of the XDM Cluster when setting up subscriptions. However, once a subscription has been established the subsequent requests will not go through the Load Balancer but rather directly to the XDMS instance hosting the subscription. All nodes in the XDM Cluster are directly accessible from the Presence Cluster.

Figure G–1 Presence and XDM Nodes



Index

M

N

O

P

S

T

U

W

X