Oracle® Communication and Mobility Server

Administrator's Guide

Release 10.1.3

E10292-02

August 2007

ORACLE®

Contents

2   Overview of Administrative Tasks

3   Deployment Topologies

4 Configuring the SIP Servlet Container

5 Provisioning Users and Applications

9 Managing the SIP Server

10 Configuring the Logging System

11 Configuring Oracle Communicator

C   The Oracle Communicator Configuration File

D   Third-Party Licensing

Index

Preface

This guide describes how to configure and manage the Oracle Communication and Mobility Server.

Intended Audience

This manual is intended for Oracle Communication and Mobility Server administrators.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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. 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 additional information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation   JAWS, a Windows screen reader, 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, JAWS 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.

Structure

This manual is organized in the following chapters:

- An Overview of Oracle Communication and Mobility Server
- Overview of Administrative Tasks
- Deployment Topologies

- Configuring the SIP Servlet Container

- Provisioning Users and Applications

- OCMS Security

- Packaging and Deploying Applications

- Configuring High Availability

- Managing the SIP Server

- Configuring the Logging System

- Configuring Oracle Communicator

- Supported Protocols, RFCs, and Standards

- Configuring Oracle Internet Directory as the User RepositoryThe Oracle Communicator Configuration File

Related Documents

For more information, see the following manuals:

- *Oracle Communication and Mobility Server Installation Guide*

- *Oracle Communication and Mobility Server Developer's Guide*

- *Oracle Communication and Mobility Server Developer's Cookbook* (available on the Oracle Technology Network at http://www.oracle.com/technology/products/ocms/otn_front.htm)

- *Oracle Communication and Mobility Server Certification Guide* (available on the Oracle Technology Network at http://www.oracle.com/technology/products/ocms/otn_front.htm)

- *Oracle Containers for J2EE Configuration and Administration Guide*

- *Oracle Containers for J2EE Deployment Guide*

- *Oracle Containers for J2EE Security Guide*

- *Oracle Internet Directory Administrator's Guide*

- *Oracle Internet Guide to Delegated Administration*

- Oracle Communication and Mobility Server resources on Oracle Technology Network (http://www.oracle.com/technology/products/ocms/otn_front.htm). This is the location for Oracle Communication and Mobility Server guides, release notes, white papers, and updates.

Support—Visit: http://www.oracle.com/support

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Convention	Meaning
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

1

An Overview of Oracle Communication and Mobility Server

This chapter provides an introduction to the Oracle Communication and Mobility Server in the following sections:

- "About SIP"
- "Introduction to OCMS"
- "OCMS Three Layer Model"
- "OCMS Deployment Modes"
- "Deployment Types"
- "OCMS System Components"

New in this Release

This release of Oracle Communication and Mobility Server includes enhancements and new features. To read about new and improved features, see the following link:

`http://www.oracle.com/technology/products/ocms/otn_front.htm`.

About SIP

The Session Initiation Protocol (SIP) is a signaling protocol for initiating, modifying, and terminating interactive communication sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. A 3GPP signaling protocol, SIP is a permanent element of the IMS architecture and is widely used for Voice over IP. Similar to HTTP, SIP is addressing neutral, with addresses expressed as URIs, telephone numbers, or addresses similar to e-mail.

Session Initiation Protocol is:

- Lightweight, with only six methods
- Transport-independent, as it can be used with UDP, TCP, and other transport protocols
- Text-based, allowing for low overhead
- Capable of handling user mobility, by registering and updating current user locations enabling delivery of messages to the correct address
- Specifically designed to work with the Internet

> **See also:** For more information on SIP, see the following URLs:
>
> - http://tools.ietf.org/html/rfc3261
> - http://www.ietf.org/html.charters/sip-charter.html

Features of SIP

SIP is distinguished by the following features:

- **Cooperative and future compatible**—SIP establishes session connectivity among users, including registering and locating users. However, SIP does not define the content of a session, such that SIP can be used to establish any type of a number of sessions. Other protocols can be used to define various sessions established using SIP, including protocols that do not yet exist.

- **End system intelligence**—SIP Servers route requests to users by examining request URIs Via headers, rather than processing data in the message body. The User Agents at either end of the system are responsible for addressing messages, such that SIP pushes intelligence to the end system. Once a session is established, end systems communicate without the help of the SIP Servers. The near statelessness of SIP Servers makes SIP a very efficient protocol.

- **Interoperability**—SIP extensions are designed to be modular, such that their use is individually negotiated between user agents. Negotiating the use of extensions guarantees interoperability among SIP user agents in the network.

- **Scalability**—Stateless SIP Servers are kept at the core of the network, while any stateful servers are kept at the periphery, near the end systems. By keeping stateless servers at the point of network stress, SIP networks are very scalable.

SIP as a Service Creation Platform

SIP reuses Internet components that are used by other Internet applications, such as elements of HTTP and SMTP. SIP both integrates and delivers services to the current, actual location of the user. SIP applications integrate well with existing Internet applications such as Web browsing, e-mail, video-conferencing, voice calls, instant messaging, and so on.

The following features make SIP an ideal choice as a platform for service creation:

- **Based on HTTP**—SIP uses a familiar text-based request/response model, and using a similar format for encoding protocol messages.

- **Uses URLS for addressing**—SIP uses URLs or e-mail addresses for addressing, enabling flexible redirection of messages.

- **SMTP-like routing**—SIP messages are routed very similarly to e-mail messages.

- **Instant messaging**—SIP is good for delivering instant messages. SIP also features a protocol for user Presence information, and a registrar server for tracking a user's current location and online status. The combination of capable instant message delivery and up-to-date management of Presence information makes SIP an ideal platform for instant messaging.

- **Forward compatible**—SIP re-purposes existing infrastructure to provide new services, for example, using a VoIP infrastructure to provide online gaming.

- **Easy to develop**—The process of developing SIP Servlet applications is very similar to the process of developing HTTP Servlet applications. SIP messages are

human readable, and SIP services can be developed without any specialized knowledge.

- **Reuses applications**—Simple SIP applications can be combined to provide complex services.

Introduction to OCMS

Oracle Communication and Mobility Server (OCMS) is a carrier-grade SIP application environment for the development, deployment, and management of SIP applications. Built on a standard Java2 Enterprise Edition (J2EE) platform, OCMS is a flexible, scalable environment enabling easy integration of SIP applications and services.

Among the applications that may be developed and deployed on SIP platforms:

- Voice and video telephony, including call management services such as call forwarding and call barring
- Publication of and subscription to user presence information, such as online/offline status, notifications, permission to access a user's status, and so on
- Instant messaging
- Push to Talk applications, including Push to Talk over Cellular (PoC)

OCMS provides standard SIP applications out-of-the-box, including a Presence Server, a combination Proxy and Registrar server, and a SIP message routing server. An integral part of any SIP platform, these applications are automatically installed to the OCMS platform, reducing development resources and time to go live.

OCMS is distinguished by its standards-based, high performance Presence Server which provides SIMPLE compliant Presence and event notification features. The OCMS Presence Server is robust enough to support carriers with a heavy load of subscribers, while still being a viable solution for ISVs, system integrators, and enterprises requiring an integration platform and an enterprise Presence Server.

OCMS provides the following functionality:

- **Development and Testing**—OCMS provides Java APIs enabling fast development of SIP applications. A SIP client and client Web Services APIs are provided for testing the SIP platform from end to end. Diameter APIs are provided to enable charging and profile lookup in the IP Multimedia Subsystem.
- **Deployment**—SIP Servlet applications are easily packaged and deployed on Oracle Application Server acting as a SIP servlet container.
- **Configuration and Management**—The Oracle Application Server administrator console enables managing SIP applications as well as the SIP server and its components. The administrator console enables managing the SIP servlet life cycle, and ensuring that users who access a URL are permitted to do so.
- **Authentication and Security**—User, role, and policy data can be stored on an Oracle TimesTen database, external RADIUS database, or in Oracle Identity Management, enabling OCMS to authenticate connecting users against this data. OCMS secures SIP traffic through digest-based authentication as specified in RFC 3261. In addition, trusted hosts can be configured using p-asserted identity headers.
- **Logging and Monitoring**—For logging and monitoring SIP applications, the OCMS provides Log4J.

OCMS Three Layer Model

Oracle Communication and Mobility Server architecture is composed of three layers:

- Proxy Layer
- Application Layer
- Data Layer

Figure 1–1   OCMS Three Layer Model
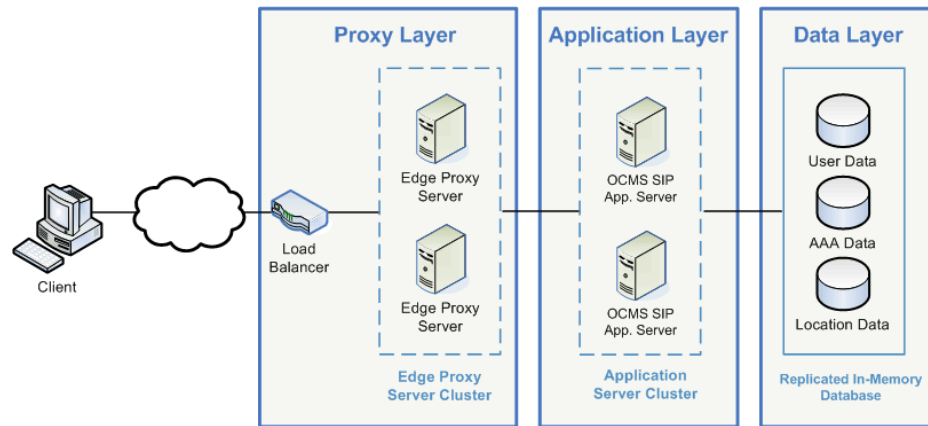
Proxy Layer

The Proxy layer includes a load balancer and the OCMS Edge Proxy Server. The load balancer provides a unique public address to which SIP requests are sent. Upon arrival, the load balancer distributes SIP requests either to the OCMS Edge Proxy Server, or directly to an OCMS SIP Server.

The SIP-unaware load balancer distributes requests to OCMS nodes based on the capacity and availability of individual servers. This is essential in a clustered environment, particularly in the event of a node failure. If a node fails, the load balancer redistributes traffic to the remaining nodes until the failure is corrected.

Most load balancers, however, are not SIP-aware, meaning that they are unaware of the content of the traffic they forward, such as the sender and recipient. The OCMS Edge Proxy provides a SIP-aware front end to a typical load balancer, proxying SIP requests to a particular OCMS SIP Server. The Edge Proxy thus forms logical pathways between sessions and SIP servers, such that SIP traffic sent from a particular session is always handled by the same server. As the number of SIP clients increases, additional Edge Proxy servers can be added, providing highly scalable and performant handling of SIP clients.

Application Layer

The Application layer is typically composed of a cluster of OCMS SIP Server nodes. The Application layer provides SIP clients with low response time and high throughput when handling SIP requests. As the Application layer handles a greater number of transactions, it can be scaled up by adding additional OCMS SIP Server nodes.

Data Layer

The Data layer is typically composed of a highly available, high performance in-memory database for the storage and retrieval of user, authentication, authorization, and location data. This data is replicated among all nodes. Similarly, SIP Servlet session data is replicated among nodes. In the event of a node failure, another node takes over the session data of the failed node.

OCMS Deployment Modes

There are two types of OCMS deployments:

- Single Node Deployment
- Clustered Deployment

For more information on configuring highly available OCMS deployments, see Chapter 3, "Deployment Topologies".

Single Node Deployment

The OCMS single node deployment runs on a single instance of OC4J application server. The application server's SIP Servlet container hosts a home-grown SIP Servlet application or combination SIP and HTTP Servlet application. A single node deployment is not highly available, and is therefore suited for development, testing, and low-volume deployments.

Clustered Deployment

A basic cluster is a configuration appropriate for small and medium scale deployments, consisting of a number of SIP Application Server instances with session state replicated among all nodes. Each SIP Application Server instance runs on a separate physical node. By increasing the number of nodes, a basic cluster provides scalability and availability. In the event of a failed node, another node takes over, minimizing downtime. A clustered OCMS deployment is easily scalable, such that adding additional nodes enables OCMS to handle a greater number of transactions.

SIP Servlet session data is replicated among nodes. This is useful in the event of a failure, where a designated node takes over the session data of the failed node.

Figure 1–2   Oracle Communication and Mobility Server Clustered Deployment
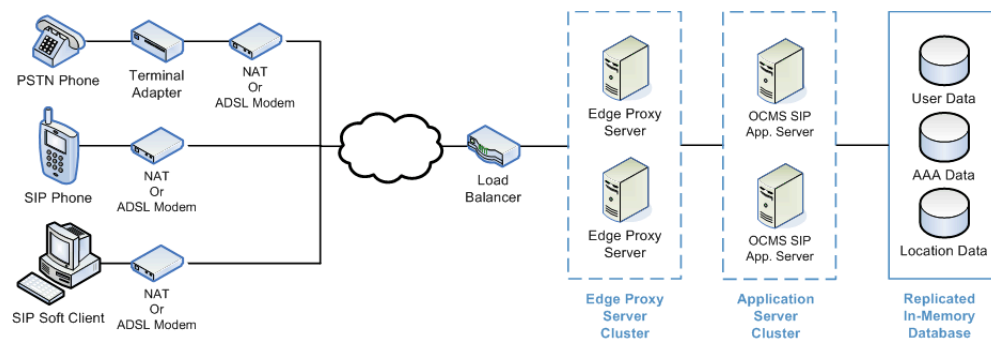
Deployment Types

This section discusses the OCMS deployment types:

■ Deploying OCMS in an IMS Network

■ Deploying OCMS as a SIP Network

Deploying OCMS in an IMS Network

OCMS is the SIP Application Server component of a 3GPP IP Multimedia Subsystem, or IMS network. An IMS network is a Next Generation Networking architecture for telecom operators providing multimedia services. IMS is an open, standards-based architecture that runs over IP, and implements a VoIP solution using SIP as the main signalling protocol. The network supports both existing packet- and circuit-switched systems as well as current and future services such as SIP applications and, presumably, any other IP-based services that exist or have yet to be developed. By using standardized, open IP protocols, IMS aims to provide seamless roaming among mobile, public WiFi and private networks for a wide range of services and devices. IMS is designed to enable operators to control and track services, allowing for time-, packet-, and service-based charging.

Figure 1–3   OCMS in an IMS Network
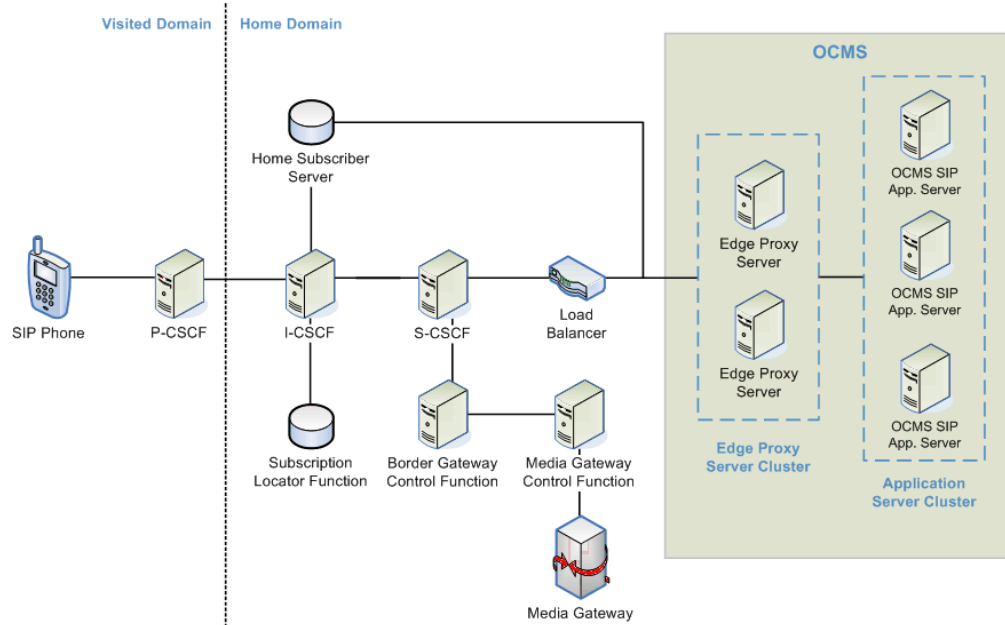
IMS Architecture

The IMS architecture is composed of SIP proxies known as Call/Session Control Functions, or CSCFs, and a Home Subscriber Server, or HSS. The main components of an IMS network are:

■ Proxy Call/Session Control Function (P-CSCF)—The P-CSCF is the first point of contact into the IMS network, forwarding all SIP requests from subscribers to the network regardless of their destination.

■ Interrogating Call/Session Control Function (I-CSCF)—The I-CSCF matches incoming subscriber requests with the correct S-CSCF using data from the Home Subscriber Server (HSS). During registration, the I-CSCF assigns subscribers to an S-CSCF.

■ Serving Call/Session Control Function (S-CSCF)—As its name suggests, the S-CSCF provides services to subscribers following registration of subscribers to an

S-CSCF. The S-CSCF handles both incoming and outgoing sessions associated with a particular subscriber.

- SIP Application Server (OCMS)—The SIP Application Server hosts SIP applications and services within the IMS network. OCMS provides a cluster of managed SIP Servlet Containers that run OCMS SIP Servlet applications such as Presence or home-grown applications.

- Home Subscriber Server (HSS)—The HSS stores information linking subscribers and their profiles with their associated S-CSCF nodes. The HSS also stores information regarding services accessible to any given subscriber, and where subscribers can be reached.

The IMS network can be divided into three planes, namely the service or application plane, the control or signalling plane, and the transport or user plane. OCMS is located in the application plane, as shown in Figure 1–4.

Figure 1–4   The Three Planes of the IMS Network
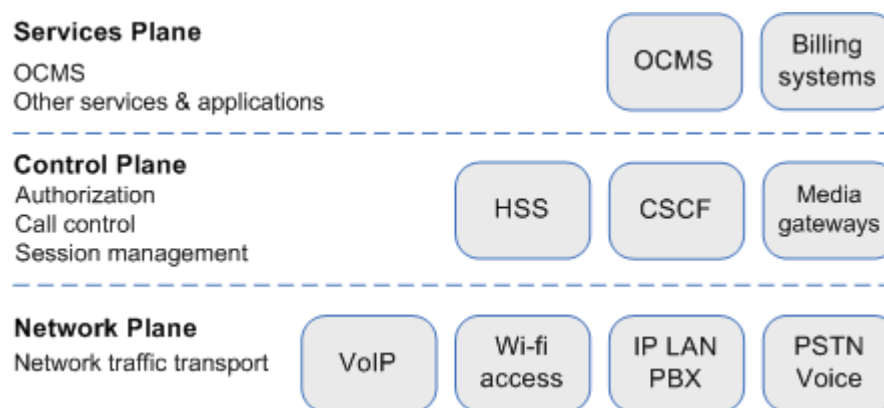
As illustrated by Figure 1–4, the IMS network uses SIP signalling to accomplish the following:

- Handling a variety of network traffic

- Authorizing subscribers and managing sessions

- Managing calls

- Providing services to subscribers

- Managing charging and billing

Typical Services Provided by a SIP Application Server within an IMS Network

Services typically provided by an IMS network include:

- Caller ID services

- Call waiting, call holding, call forwarding, call transfer

- Call blocking services

- Push to talk

- Lawful interception

- Announcement services

- Conference call services

- Voicemail, text-to-speech, speech-to-text

- Location based services

- SMS, MMS

- Presence information, instant messaging

OCMS in the Context of an IMS Network

When used in the context of an IMS network, OCMS acts as the service layer within the IMS network, hosting SIP applications that provide services to end users. The OCMS Edge Proxy distributes incoming SIP requests to OCMS nodes, assigning each request to a particular node. Each OCMS node handles incoming SIP requests and outgoing SIP responses. Outgoing responses return to the correct Edge Proxy, which sends SIP traffic to the IMS core network. The IMS core network, in turn, sends responses back to the end users.

Figure 1–5   OCMS as Part of an IMS Network
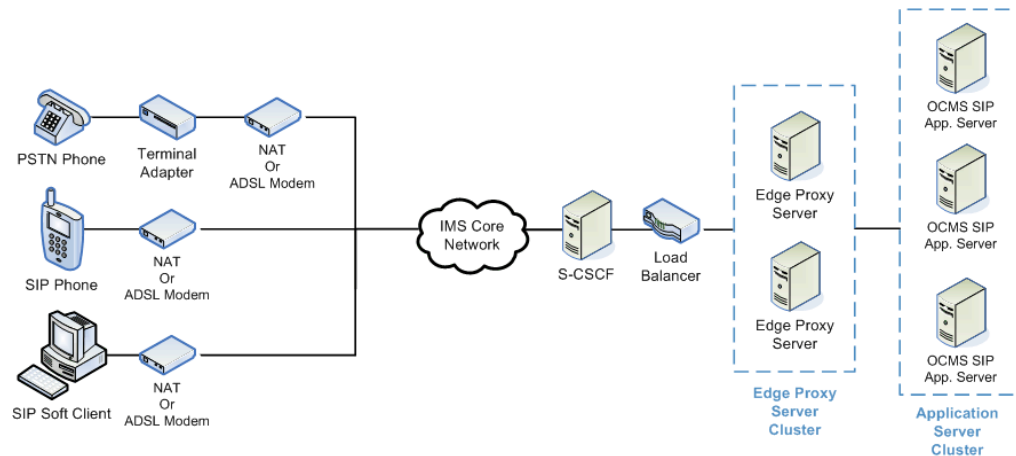
Deploying OCMS as a SIP Network

Another way to deploy OCMS is in place of a vanilla SIP network. In this scenario, OCMS comprises the SIP network itself. As shown in Figure 1–6, is composed of three main layers:

- Proxy layer—Including a SIP-unaware load balancer and one or more Edge Proxy servers.

- Application layer—Including one or more OCMS application servers running a SIP Servlet container. The SIP Servlet container manages SIP applications such as the OCMS Presence Server, Proxy/Registrar, Application Router, and so on. Other home-grown SIP and merged SIP/HTTP applications also run on the SIP Servlet container.

- Data layer—Includes a replicated, in-memory database where user, authentication, and location data are stored and accessed. These data are used to register and authenticate users, as well as deliver SIP messages to the correct destination.

Figure 1–6   OCMS Deployed as a SIP Network
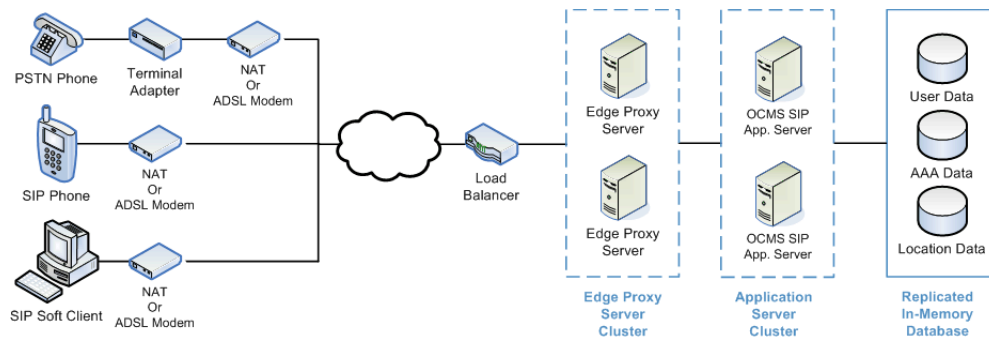
When deployed as a SIP network, OCMS has much the same functionality as a full-scale IMS network, but on a smaller scale and at a lesser cost. Deploying OCMS as a SIP network enables the following:

- Handling SIP traffic originating from a variety of clients, including PCs, PSTN phones, SIP-enabled mobile phones.

- Proxying incoming SIP messages to OCMS application servers, and other SIP enabled nodes such as media servers and other SIP proxies

- Managing user or session connections to specific application servers in a clustered OCMS environment

- Registering user data and location, storing user profiles

- Routing SIP messages through a chain of applications, and ensuring that messages arrive at the correct destination

- Registering, storing, and retrieving user presence information

OCMS System Components

Figure 1–7 illustrates the logical system components of OCMS:

Figure 1–7   Oracle Communication and Mobility Server
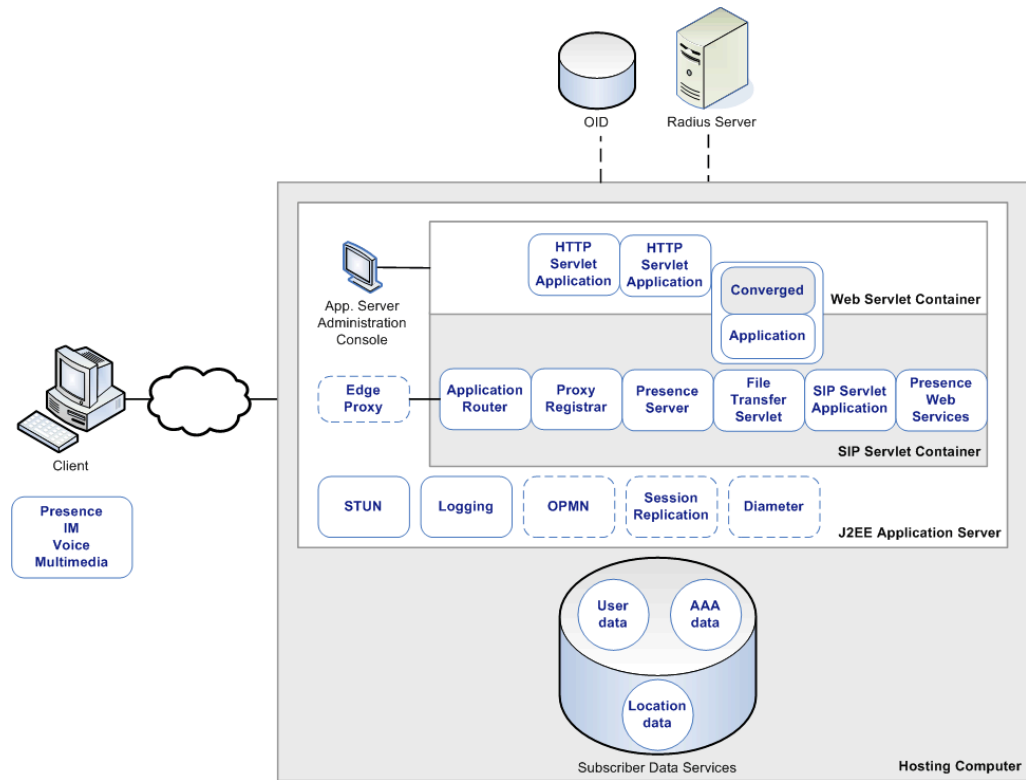
The OCMS components are as follows:

- SIP Servlets and SIP Servlet Applications

- SIP Servlet Container

- Edge Proxy Server

- Proxy Registrar

- Presence Server

- Application Router

- Diameter

- Subscriber Data Services

- Logging

- Session Replication

SIP Servlets and SIP Servlet Applications

Servlets are dynamic applications that run on a web server using the J2EE platform. Like the HTTP Servlet API, the SIP Servlet API (JSR116) extends the functionality of the Java Servlet to receive SIP requests and generate SIP responses, regardless of the underlying network. A SIP application consists of one or more SIP Servlets which, along with a deployment descriptor, are packaged and deployed on a J2EE SIP Servlet Container.

Differences between HTTP and SIP Servlets

Although they are similar, the SIP Servlet differs from the HTTP Servlet as follows:

- SIP applications include intelligent request routing and the ability to proxy requests as required, even to multiple destinations.

- SIP is a peer-to-peer protocol, with endpoints that can typically initiate and respond to SIP requests.

- SIP Servlet applications may be registered so as to be invoked in response to particular events.

- Unlike the HTTP Servlet, the SIP Servlet is asynchronous. This means that when receiving a SIP request, a SIP Servlet application can initiate another action, return control to the SIP Servlet container, and respond to the request at a later time.

- A SIP Servlet application is often composed of more than one SIP Servlet.

- As SIP and HTTP servlets are based on the same generic Servlet specification, both types of servlets can be easily converged into one application. An application composed of both SIP and HTTP servlets can therefore handle both SIP and HTTP traffic.

Figure 1–8   SIP Servlet Model versus HTTP Servlet Model
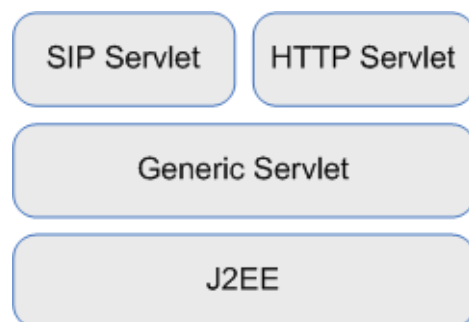
Typical SIP Servlet Applications

Typical SIP-based applications include:

- Telephony over IP, with the following features:
    - Speed dial
    - Wake-up call service
    - Call forwarding service
    - Click-to-call
    - Emergency call service
- Video calls
- Push to talk
- Instant messaging
- Presence information service
- Network gaming

Figure 1–9   SIP Servlet Applications
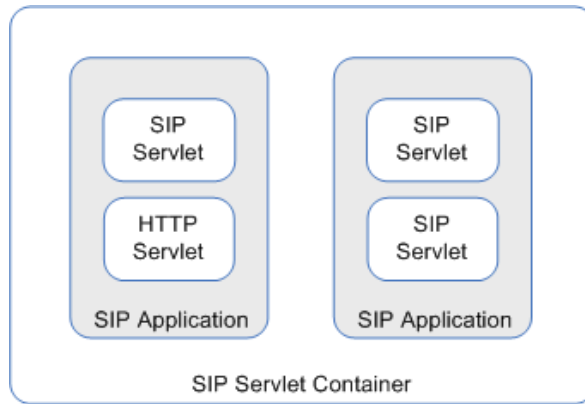
SIP Servlet Container

A SIP Servlet Container extends the J2EE Application Server, providing a runtime environment for SIP applications, including services such as security, concurrency, life cycle management, transaction, deployment, and other services. A JSR116-compliant SIP Servlet Container provides network services for sending and receiving SIP requests and responses using a combination of transport protocols, IP addresses, and port numbers to listen for incoming SIP traffic.

The OCMS SIP Servlet Container can be installed on an existing instance of Oracle Application Server, running in OC4J. Alternatively, the OCMS SIP Servlet Container can run on its own stand-alone instance of OC4J.

The typical OCMS SIP Servlet Container is composed of an Oracle Application Server instance with OC4J as its J2EE container, and Oracle Process Manager and Notification Server (OPMN) to monitor the server. OCMS currently supports high availability deployments in this configuration only.

Figure 1–10   The SIP Servlet Container on Oracle Application Server
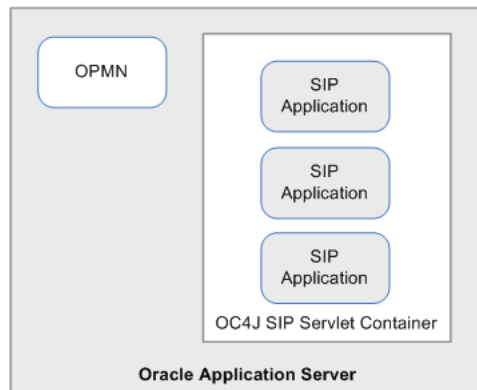
How the OCMS SIP Servlet Container Works

The SIP Servlet Container is configured upon server startup. Once a SIP Servlet application is deployed to the SIP Servlet Container, its deployment descriptor is used to configure its servlets and create a servlet context. The SIP application's listeners and servlets are instantiated, and the servlets are initialized with the servlet configurations.

When the SIP Servlet Container receives an initial incoming request, it processes a set of rules in order to send the request to the correct SIP Servlet. Once the request arrives

at the SIP Servlet, the Servlet must either proxy the request to a new destination, dispatch it to another Servlet, or send a response.

Edge Proxy Server

The Edge Proxy server provides the following functionality:

- Acts as a load balancer for initial incoming SIP requests

- Provides SIP Server affinity and failover for subsequent SIP requests in a session

- Manages the health of the OCMS application servers in a cluster by dynamically constructing a routing table of OCMS application servers

The Edge Proxy distributes incoming SIP traffic among OCMS SIP application servers when used between a SIP-unaware load balancer and an OCMS cluster. A standalone Java application running on its own server, the Edge Proxy establishes an affinity between the client and SIP Application Server for the duration of the session. This means that the same OCMS SIP Application Server always handles traffic from a particular client for the duration of the session, creating a path between the client and server.

Figure 1–11   Edge Proxy Functionality
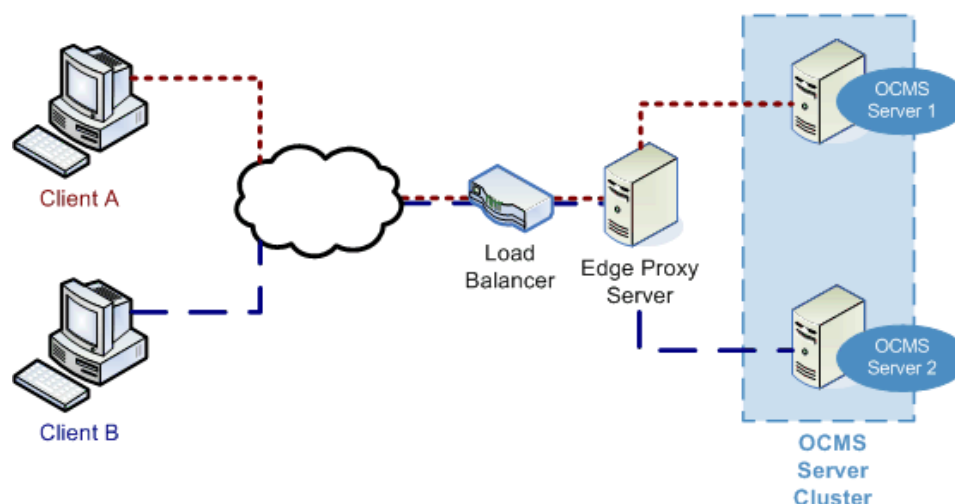
Multiple Edge Proxy Servers can be deployed in a highly available environment. In this scenario, a load balancer distributes incoming traffic among the Edge Proxy Servers. Together, the Edge Proxy Servers can handle a greater load of subscribers connecting to the cluster of OCMS SIP Application Servers.

Figure 1–12   Multiple Edge Proxy Servers
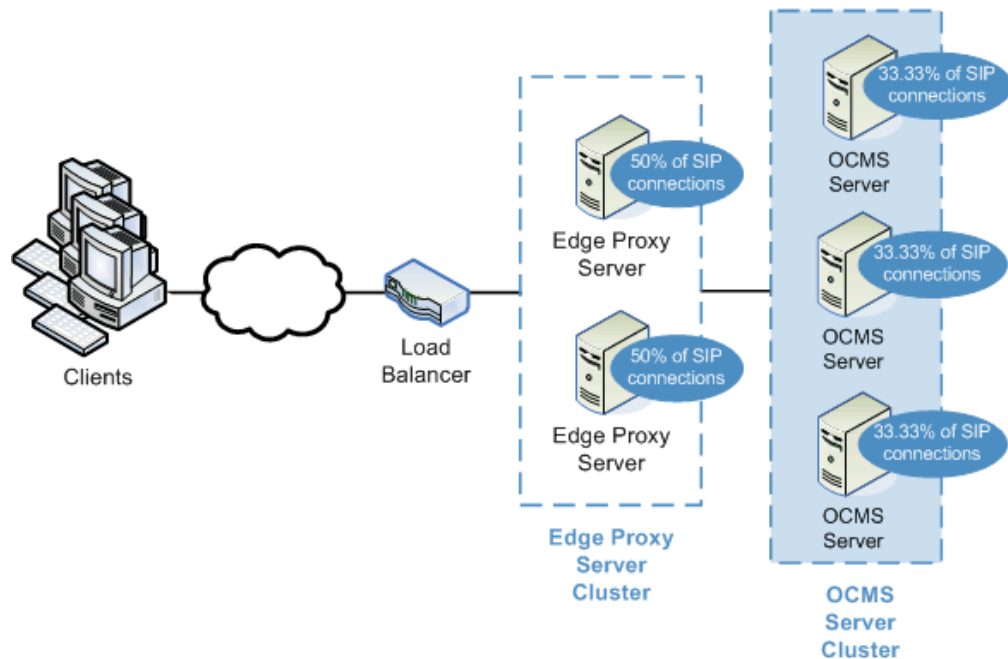
Figure 1–12 illustrates how the use of two Edge Proxy Servers reduces the load of SIP connections in a clustered OCMS environment. A third-party load balancer provides a single virtual IP address to which clients may address requests, and distributes SIP requests to the Edge Proxy Servers. Edge Proxy Servers can be duplicated to enable high availability—if one Edge Proxy fails, the other Edge Proxy takes over the workload of the failed node. When scaling up OCMS Server nodes, it may be necessary to add additional Edge Proxy Servers to the topology in order to handle the additional connections being established in the system.

Proxy Registrar

The OCMS Proxy Registrar combines the functionality of a SIP Proxy Server and Registrar. Its main tasks include:

- **Registering subscribers**. The Proxy Registrar registers a subscriber's address and maps it to the actual address of the subscriber's terminal. The Proxy Registrar stores subscriber contact information in the Location Service data store, and uses this data to create paths between the SIP Application Server and the subscriber.

- **Proxying requests onward**. Upon receiving SIP requests, the Proxy Registrar finds the current contact information of the subscriber using the Location Lookup Service or ENUM (*TElephone NUmber Mapping*) Service. The Proxy Registrar replaces the request destination URI with the current, correct SIP address as returned by one of the lookup services, and proxies the request to this destination.

Figure 1–13 Proxy Registrar
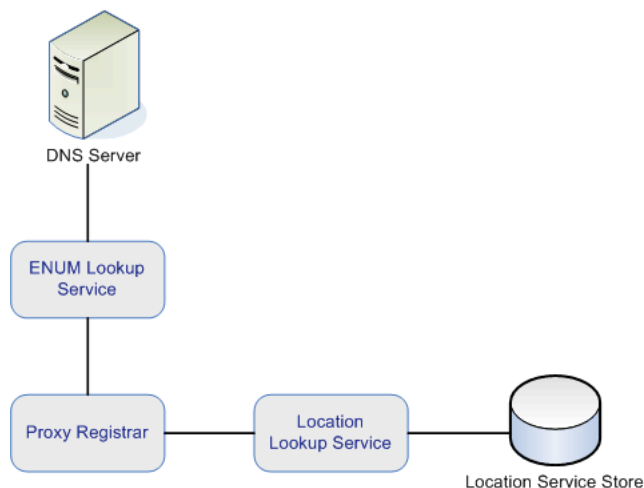
Location Lookup Service

The Location Lookup Service stores registration information for all subscribers, as defined by RFC3261. This information is used by the Proxy Registrar to reach subscribers at the right client at any time. Registration data is stored in an Oracle TimesTen In-Memory Database (*Oracle TimesTen*) database. A given subscriber might connect to OCMS using a client at home or work, for example. The Proxy Registrar uses the Location Service to look up the subscriber's actual, current contact information and proxy requests to that URI. The Proxy Registrar thus creates a direct, reusable connection between the user and the node. The Proxy Registrar uses this connection to route subsequent requests to the correct destination. The client, meanwhile, must regularly refresh its state so as to keep the Location Service data current.

ENUM Lookup Service

If an incoming SIP request destination URI includes a telephone URI, the Proxy Registrar must translate the telephone number to a SIP address, using an ENUM (*TElephone NUmber Mapping*) service. OCMS provides an ENUM Service which uses a configured DNS server to look up the telephone number and translate it into a SIP address. The ENUM Service replaces the telephone number destination URI with the translated SIP address and proxies the request to its destination.

The main tasks of the ENUM Lookup Services are as follows:

1. Convert the telephone destination in an incoming request URI into a host name. For example:

    ```
    $ORIGIN 2.4.2.4.5.5.5.5.1.4.1.e164.arpa.
    ```

2. Look up the converted host name in the configured DNS server. The DNS server returns a matching SIP address based on its search for the phone number.

    ```
    IN NAPTR 10 100 "u" "E2U+sip" "!^.*$!sip:4242@555telco.example.com!"
    ```

3. Replace the telephone number URI with a properly formatted SIP address.

4. Proxy the request to the SIP address.

Presence Server

OCMS includes its own Presence Server, based on the following: RFCs 2778, 3265, 3856, 3857, 3858, 3859, 3863, 3903, as well as OMA Presence Enabler 1.0. The OCMS Presence Server handles registration, storage, and retrieval of presence information.

Presence describes a user's availability and willingness to communicate. The Presence Server can signal whether users are on- or offline and whether they are idle or available. The Presence Server enables users publish their contact details such as instant messaging handle, mobile phone number, audio and video capability, and so on.

The Presence Server does the following:

- Processes presence PUBLISH requests

- Composites event state into a presence document for a presentity

- Accepts SUBSCRIBE requests from watchers to create subscriptions to a given presentity's presence data

- Acts as a notifier, generating NOTIFY requests to notify subscribers of the state of their subscribed presentity

A number of roles are defined in the context of Presence:

- Presentity—A Presentity is a user entity that provides presence information to the Presence Server. A presentity can have a number of PUAs, such as a computer at work, a computer at home, and a mobile device.

- Presence User Agent (*PUA*)—A device that provides presence information, such as an instant messaging application (Oracle Communicator), or a mobile device. A PUA provides presentity information to the Presence Server.

- Watcher—Requests presence or watcher information from the Presence Server. The two types of watchers include the fetcher and subscriber.

  - Fetcher—Retrieves a presentity's presence data from the Presence Server.

  - Subscriber—Subscribes to a presentity's presence information, so as to be updated with current information regarding that presentity's presence.

Figure 1–14   Basic Presence Functionality
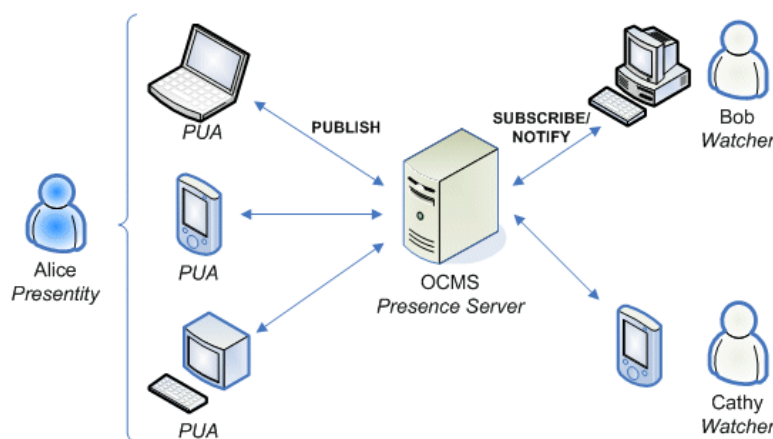
As illustrated in Figure 1–14, presentity Alice has three Presence User Agents (PUAs). A client running on any of these PUAs publishes Alice's presence to the OCMS

Presence Server. Meanwhile, watchers Bob and Cathy want to subscribe to Alice's presence information.

Bob and Cathy each run a client that sends the Presence Server a SUBSCRIBE request for Alice's presence. The Presence Server consults Alice's presence policy document in order to determine if Bob and Cathy are permitted to subscribe to his presence. If they are, then the Presence server sends a NOTIFY message containing information about Alice's current presence state. Whenever Alice's presence state changes, the Presence Server sends a NOTIFY message to Bob and Cathy's clients informing them of the change

How the Presence Server Works

The following example illustrates how the Presence Server manages presence information.

Figure 1–15   How the Presence Server Works
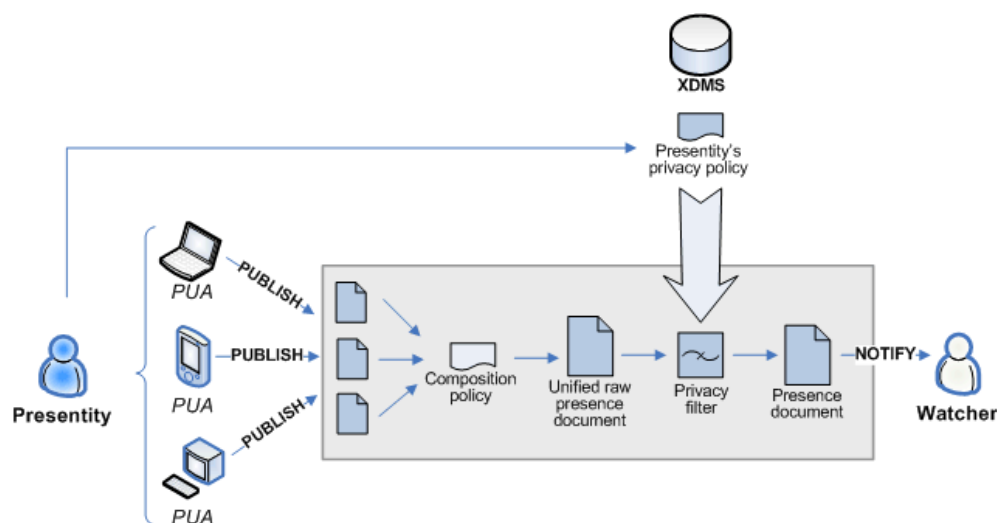
Suppose a user's presence information has changed. For example, the user might be using a different PUA such as a mobile device or a laptop, or the user may be idle or away. The following describes how the Presence Server handles this change in data:

1. The presentity uploads a policy document that specifies the information to which each watcher is entitled. For example, a user might only want particular watchers to see whether or not she is online.

2. Each PUA sends the new presence information to the Presence Server and issues a SIP PUBLISH request. The presence information is sent in the form of a presence document.

3. The Presence Server receives the presence documents and merges the data into a single document using a composition policy that specifies rules regarding merging presence documents.

4. The unified document is filtered using the privacy policy uploaded to the Presence Server by the presentity (see step 1). This filtering removes any details that the presentity does not want to provide to a given watcher.

5. The Presence Server sends the watcher a NOTIFY request containing the presence document.

Application Router

OCMS 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.

The Application Router typically routes SIP requests to the Proxy Registrar or the Presence Server, respectively. Any number of additional application URIs can be configured in the Application Router. For more information on configuring the Application Router, see "Configuring the Application Router".

Modes of Operation

The Application Router operates in two modes: standard and incremental.

Standard Mode The Application Router embeds any number of destination URIs, or routes, in the headers of incoming SIP requests. The SIP request follows this chain of URIs until the request is consumed. The order of URIs is determined by the incremented alias assigned to each route (`uri.1`, `uri.2`, and so on).

Figure 1–16   The Application Router in Standard mode
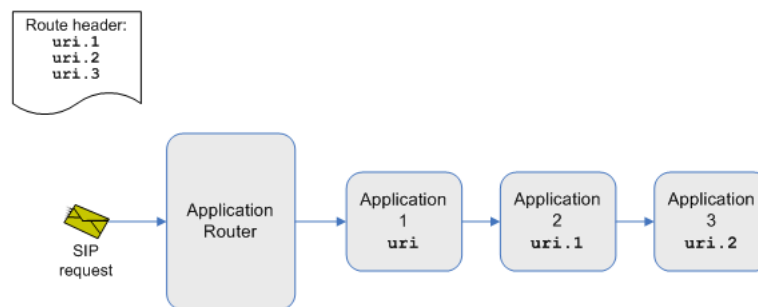
Incremental Mode The Application Router incrementally embeds each route within the incoming SIP request, along with a route back to the Application Router. The SIP request is sent to the first destination, and then returns to the Application Router. The Application Router examines the SIP request's destination URI, which results in one of two possible outcomes:

- If the destination URI has been changed by the application to which the request was sent, the Application Router proxies the SIP request to the new URI.

- If the destination URI has not changed, the Application Router embeds the second route in the header of the SIP request and sends it on its way. Again, the SIP request arrives at the second destination, whereupon it returns to the Application Router. The Application Router must, once again, decide whether to proxy the SIP request to a new URI or embed the third route in the header of the SIP request.

For example, as shown in the following illustration, the Application Router embeds within a SIP request destinations `uri.1`, `uri.2`, and `uri.3`. The SIP request goes first to `uri.1`. If application 1 changes the destination URI of the SIP request, the Application Router routes the request to the new URI, or application x (see Figure 1–17). Otherwise, the Application Router routes the request to the next URI configured in the route header, namely `uri.2`.

Here again, application 2 may change the destination URI of the SIP request, in which case the SIP request continues on to application y (see Figure 1–17). Otherwise, the Application Router routes the SIP request to uri.3, and so on.

Figure 1–17   The Application Router in Incremental Mode
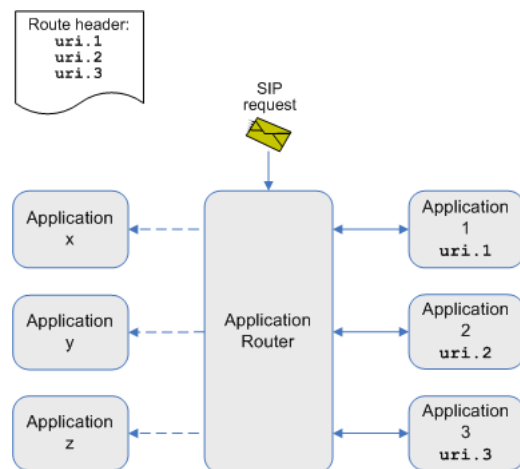
Using the Application Router in Standard Mode: an Example

The Application Router can be used in conjunction with a call screening application, for example. In this scenario, the Application Router runs in standard mode with two destination URIs configured in the route header: one to the call screening application and one to the OCMS Proxy Registrar.

An incoming SIP request is intercepted by the Application Router, which embeds both the call screening application URI and the Proxy Registrar URI in the route header of the SIP request. The SIP request continues on to the call screening application, which determines whether or not to put through the request for a call to a given user.

If the call screening application accepts the call, the SIP request continues on to the Proxy Registrar, which forwards the request to the correct destination.

If the call screening application rejects the call, it responds with a "403 Forbidden" error message for example, stopping the SIP request and breaking the routing chain.

Using the Application Router in Incremental Mode: an Example

The Application Router can be used in the context of a call forwarding application. A call forwarding application typically forwards calls by modifying the SIP request destination URI. For example, a call forwarding application might change the destination URI to the URI of a voice mail server.

This is accomplished by using the Application Router in incremental mode to intercept and proxy the SIP requests. SIP requests are sent to the Application Router, which forwards the requests to the call forwarding application and places a return route to itself in the header of the SIP request. The call forwarding application determines whether or not to send the SIP request to the voice mail server and consequently modifies the SIP request destination URI. As the SIP request destination URI has changed, the SIP request returns to the Application Router, which proxies the SIP request to the destination determined by the call forwarding application.

Suppose the SIP request returns to the Application Router, but the call forwarding application has not changed its destination URI. In this case, the Application Router

sends the SIP request to the next application as configured in the Application Router settings.

Diameter

Diameter is used for charging and profile lookup (*Home Subscriber Server*) in the IMS.

The Diameter protocol provides the following features:

- Connection and session management
- Reliable delivery of attribute value pairs (AVPs)
- Agent support for proxy, redirect, and relay servers
- APIs for developing applications such as user authentication, configuration of permissions, and basic accounting services

Oracle's Diameter component includes the following IMS interfaces:

- Sh
- Ro and Rf

Sh

The Sh interface operates between the SIP Application Server and the HSS network elements in the IMS. The Sh interface enables:

- Downloading and updating user data
- Requesting and sending notifications when changes are made to user data

Ro and Rf

Diameter provides online and offline interfaces for online billing or charging, called Ro and Rf, respectively.

Ro is used for charging services in real-time (as the service occurs). Based on the IETF Credit Control Application (RFC 4006), the Ro interface uses the Credit-Control command (CCR/CCA).

Rf is used to transfer charging data that has no affect on services being rendered in real-time. The Rf interface is based on the accounting functionality of the IETF-diameter base (RFC 3588). Rf uses the accounting command (ACR/ACA).

Subscriber Data Services

The OCMS subscriber data services stores user authentication data, user, role, and policy data, as well as user location data. In a scaled, highly available configuration, the shared database enables all OCMS SIP application servers to access stored data.

The following data is stored:

- Authentication and Authorization Data
- User Data
- Location Lookup Data

Authentication and Authorization Data

Authentication and authorization data provide the primary framework through which access control is configured for the OCMS. Authentication and authorization data can

be stored on an Oracle TimesTen database, RADIUS server, or on Oracle Identity Management.

User Data

The User database stores private subscriber IDs used for subscriber authentication. The Proxy Registrar authenticates users by mapping private subscriber IDs stored in the user database to public subscriber addresses in SIP requests and responses. User data is stored on an Oracle TimesTen database or Oracle Identity Management.

Location Lookup Data

Location data is always stored in a persistent or in-memory Oracle TimesTen database. The TimesTen database persists Location data beyond a server restart.

Logging

The Logging service is used to log and monitor system events and SIP traffic. Logs can be used to audit and debug the system.

Session Replication

The Session Replication module handles the replication of session state among multiple SIP Application server nodes. Built on top of OC4J clusters, Session Replication is used only in high availability scenarios.

2

Overview of Administrative Tasks

This chapter discusses the main flow of administrative tasks required to manage the OCMS in the following topics:

- "Overview"
- "Installing OCMS"
- "Deployment Topologies"
- "Provisioning"
- "Configuring the SIP Servlet Container"
- "Managing Authentication and Security"
- "Deploying Applications"
- "Configuring High Availability (HA)"
- "Managing OCMS Components"
- "Logging"
- "Troubleshooting"

Overview

This chapter describes the main flow of administrative tasks required to configure and manage OCMS.

Installing OCMS

For installation instructions, see *Oracle Communication and Mobility Server Installation Guide*.

Deployment Topologies

Refer to Chapter 3, "Deployment Topologies" for descriptions of the following:

- OCMS components
- Highly available OCMS topologies
- Single node OCMS topologies
- Choosing an appropriate topology

> **See:** For more information , see "Configuration Recommendations". See also Chapter 8, "Configuring High Availability".

Provisioning

Provisioning users and applications involves:

- Creating users
- Assigning users communication IDs and security roles
- Provisioning XDMS application usage

> **See:** For more information, see Chapter 5, "Provisioning Users and Applications".

Configuring the SIP Servlet Container

Configuring the SIP servlet container involves:

- Managing the SIP servlet container
- Configuring the following:
  - The IP address and port from which the SIP servlet receives traffic
  - The domains and realms handled by the SIP servlet
  - The DNS caching server to be used by the SIP servlet container
  - The contact address to embed in the headers of outgoing SIP messages
  - The Application Router, used to route SIP requests to the correct applications.

> **See:** For more information, see Chapter 4, "Configuring the SIP Servlet Container".

Managing Authentication and Security

Managing OCMS authentication and security involves:

- Configuring user authentication
- Authorizing Presence
- Securing traffic with digest authentication and trusted hosts

> **See:** For more information, see Chapter 6, "OCMS Security".

Deploying Applications

Deploying applications involves packaging developed applications for deployment on the SIP server. The main steps of deploying an application include:

- Packaging an application
- Deploying an application
- Starting an application

> **See:** For more information, see Chapter 7, "Packaging and Deploying Applications".

Configuring High Availability (HA)

Configuring high availability involves:

- Configuring OCMS SIP containers for high availability
- Configuring SIP Servlet applications for high availability
- Configuring the Edge Proxy for high availability
- Configuring the Proxy Registrar for high availability

> **See:** For more information, see Chapter 8, "Configuring High Availability".

Managing OCMS Components

Managing OCMS components involves:

- Managing components using the Oracle Application Server Control Console
- Configuring deployed applications
- Monitoring performance

> **See:** For more information, see Chapter 9, "Managing the SIP Server".

Logging

Logging SIP traffic involves:

- Setting log levels
- Viewing and interpreting log data, including the following categories:
  - Bad Message logs
  - Configuration logs
  - Traffic logs
  - System logs
  - Charging/event logs
  - Statistics logs
- Managing logging

> **See:** For more information, see Chapter 10, "Configuring the Logging System".

Troubleshooting

The troubleshooting chapter presents possible problems and solutions, along with a list of known issues and workarounds.

3

Deployment Topologies

This chapter discusses OCMS deployment topologies in the following sections:

About Deployment Topologies

OCMS supports two main categories of deployment topologies: single node and clustered.

A single node deployment consists of a single SIP Application Server instances running on one computer. Such a deployment typically runs one or two SIP applications along with an in-memory database. A single node deployment is therefore appropriate for running a testing environment or a very small deployment of OCMS.

A SIP Application Server cluster is defined as a set of SIP Application Server instances that share state related to the applications. A cluster includes one or more application server nodes, with each node running one instance.

A highly available OCMS cluster provides the following:

- Replication of objects and values contained in a SIP Application Session.

- Replication of location service data.

- Load balancing of incoming requests across OCMS SIP application servers.

- Overload protection keeps internal queues short by rejecting new transactions when necessary, and handling new transactions first when in overload mode.

- Transparent failover across applications within the cluster, including application restart which enables applications that fail mid-session to restart and continue execution on another instance.

Table 3–1   Additional Information

For more information on...	See:
OCMS installation	*Oracle Communication and Mobility Server Installation Guide*
Operating systems supported by highly available OCMS clusters	*Oracle Communication and Mobility Server Certification Guide*
Configuring a highly available clustered Oracle Application Server environment	- The "Application Clustering" chapter in *Containers for J2EE Configuration and Administration Guide*. - The "Active-Active Topologies" chapter in *Oracle Application Server High Availability Guide*.
Configuring highly available OCMS topologies	Chapter 8, "Configuring High Availability" in this guide

Topology Components

Components of a highly available topology include the following:

- Third-Party Load Balancer
- Edge Proxy Nodes
- SIP Application Servers
- Oracle TimesTen In-Memory Database
- Aggregation Proxy
- Proxy Registrar

Third-Party Load Balancer

A third-party load balancer balances the load of incoming traffic among the Edge Proxy nodes. It also deflects failed Edge Proxy nodes and redirects traffic to other Edge Proxy nodes.

A third-party load balancer is an optional component in most topologies. However, it is useful in an IMS topology.

Edge Proxy Nodes

The Edge Proxy nodes form sticky connections between clients and servers for the duration of the session, creating a path between a client and server and sending SIP traffic over that path. The Edge Proxy nodes balance the load of SIP traffic among the SIP Application Servers.

Both Edge Proxy Servers are configured with a virtual IP address. Each Edge Proxy node detects failed SIP Application Server nodes and fails over to the remaining SIP Application Server nodes.

SIP Application Servers

The SIP Application Servers are all linked to each Edge Proxy node. The SIP Application Servers are linked to each other through OPMN. The OCMS SIP state on each computer is replicated to the other two nodes. If one SIP Application Server node fails, another node takes over, using the replicated state of the failed node.

For more information on replicating states among OAS instances and configuring clustering, see "Setting Up a Highly Available Cluster of OCMS Nodes".

Oracle TimesTen In-Memory Database

An Oracle TimesTen database runs on each SIP Application Server node in the relevant topologies. Each database is synchronously replicated to the other. If a SIP Application Server node fails, the other node takes over using the replicated database transactions of the failed node.

An Oracle TimesTen database is required for the following OCMS components:

- Proxy Registrar

- Presence Server

- Any user-defined applications that use a database

For more information on configuring replication for Oracle TimesTen In-Memory database instances, see "Configuring the Proxy Registrar for High Availability" on page 8-13.

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 Server. This is an optional component.

Proxy Registrar

The OCMS 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 TimesTen database. This is an optional component.

For more information, see "Proxy Registrar".

Supported OCMS Topologies

Supported topologies include:

- Deploying OCMS as a SIP Application Server

- Deploying OCMS as a Highly Available SIP Network

- Deploying OCMS as a Presence Server

- Deploying OCMS as an Instant Messaging Platform

- Deploying an OCMS Testing Environment

> **Note:** Only the Oracle Application Server installation mode supports high availability. For more information, refer to the *Oracle Communication and Mobility Server Installation Guide*.

Deploying OCMS as a SIP Application Server

OCMS may be deployed as a highly available cluster of SIP Application Servers. This topology may be deployed as part of a IMS network, with OCMS providing highly

available access to SIP-based services. This topology is useful to carriers wishing to host SIP Servlet applications, and may include the following components:

- A hardware load balancer (optional)

  Alternatively, the S-CSCF can balance the load in this topology using one of its supported methods.

- Two Edge Proxy nodes

- Three SIP Application Server nodes

Figure 3–1   OCMS as a SIP Application Server: Five Node SIP Application Server Topology
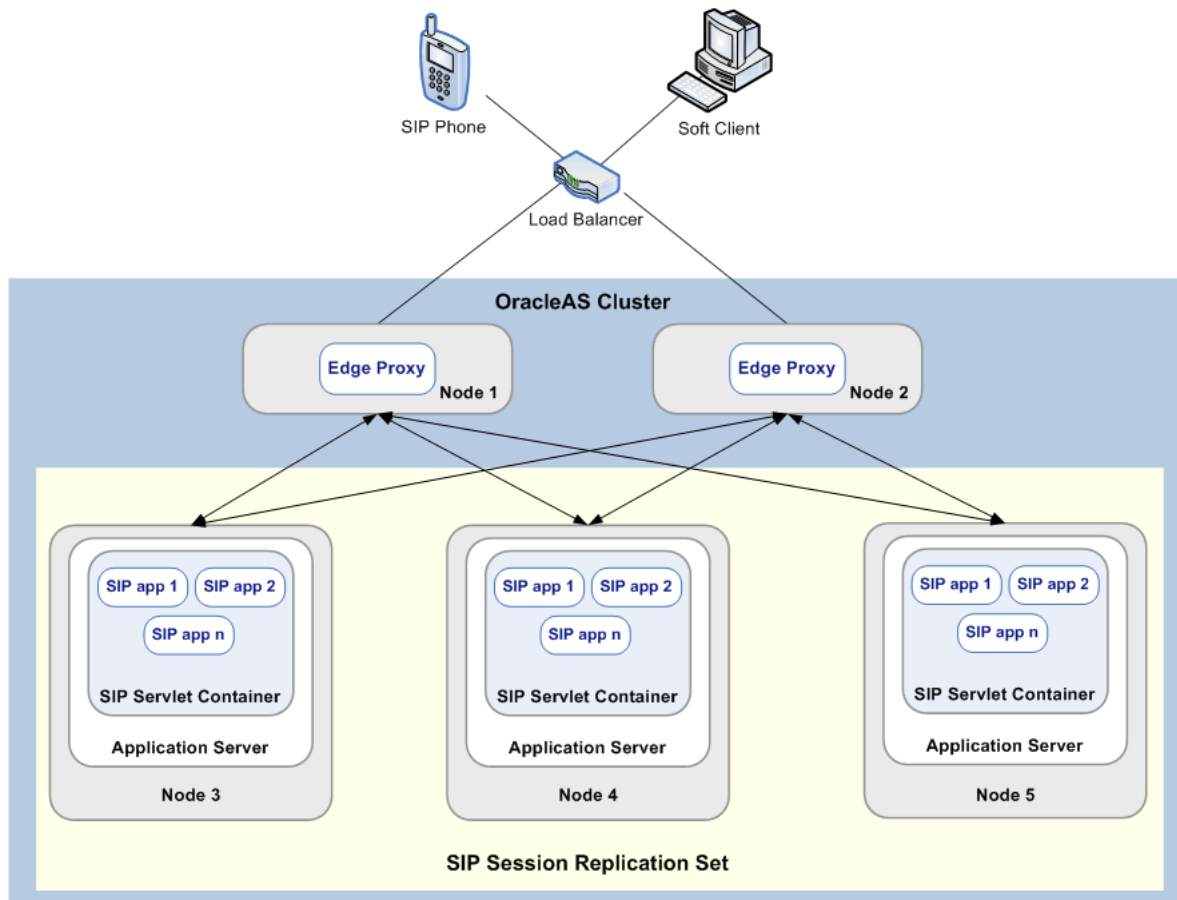
The SIP Application Server cluster topology does not include authentication or authorization mechanisms. When used in an IMS network, the S-CSCF node typically handles authentication for the SIP Application Server cluster. Applications deployed to this topology must follow certain guidelines in order to support high availability (see "Configuring High Availability"for details).

The OCMS as a SIP application network topology includes hardware and software components described in Table 3–2.

Table 3–2   OCMS as a SIP Application Network Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
Load balancer		
Two computers with at least 4 GB of RAM and a dual 2.8 Ghz CPU	Edge Proxy	Custom installation
Three computers with 4 GB of RAM and a dual 2 Ghz CPU	OAS 10.1.3.2	Typical installation

[1]   Refer to the *OCMS Installation Guide* for more information.

> **See also:**   "Topology Components" for a description of the components used in this topology.

Deploying OCMS as a Highly Available SIP Network

When deployed as a highly available SIP network, OCMS can be used to implement a basic VoIP system, enabling voice and video calls. This topology includes the following:

- A hardware load balancer

- A cluster of Edge Proxy Servers

- A cluster of two SIP Application Servers, each running a SIP Servlet Container

- Replicated in-memory databases, including user data, authentication data, and user location information

Figure 3–2 OCMS as a Highly Available SIP Network

**Four Node OCMS SIP Application Server Cluster
with Proxy Registrar**
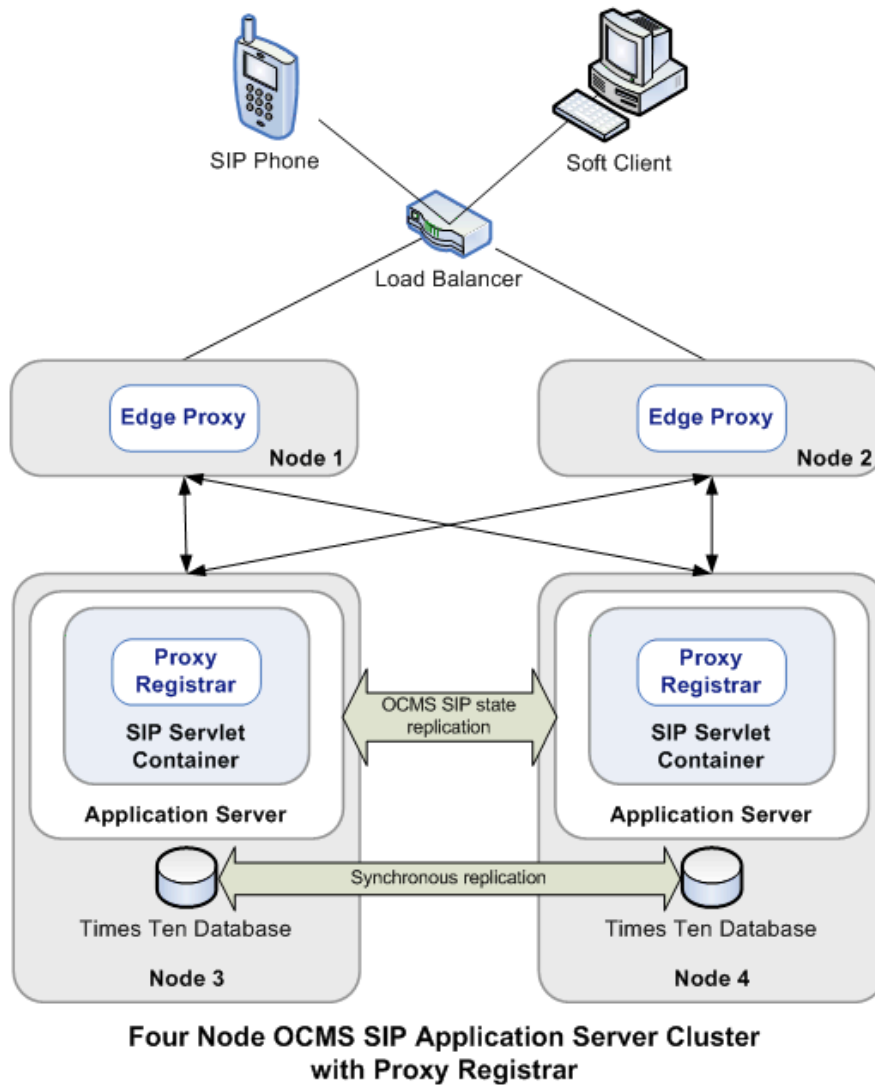
This topology provides a highly available SIP network capable of handling up to one million users[1]. Each SIP Application Server must run an Oracle TimesTen database and Proxy Registrar application.

The SIP network topology includes hardware and software components described in Table 3–3.

Table 3–3 SIP Network Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
Load balancer		
Two computers with at least 4 GB of RAM and a dual 2.8 Ghz CPU	Edge Proxy	Custom installation

---

[1] See Oracle "Reference Call Model" for more information.

Table 3–3   (Cont.) SIP Network Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
Two computers with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ Oracle TimesTen database ■ Proxy Registrar Application	Typical installation

[1] Refer to the *OCMS Installation Guide* for more information.

Load Balancer

A load balancer or DNS round-robin algorithm balances the load of incoming traffic among the Edge Proxy nodes. Using the DNS round-robin algorithm requires all clients to implement DNS lookup.

> **See also:**   "Topology Components" for a description of the components used in this topology.

Deploying OCMS as a Presence Server

OCMS can be deployed as a Presence Server. The Presence Server topology is deployed on two nodes, one running the Presence Server, the other running the Aggregation Proxy and XDMS Server. This topology can be implemented within an IMS network in order to provide Presence functionality.

The Presence Server topology includes the following:

■   One SIP Application Server node with the Presence application

■   One SIP Application Server node with an XDMS server and an Aggregation Proxy

Figure 3–3   Presence Server Topology
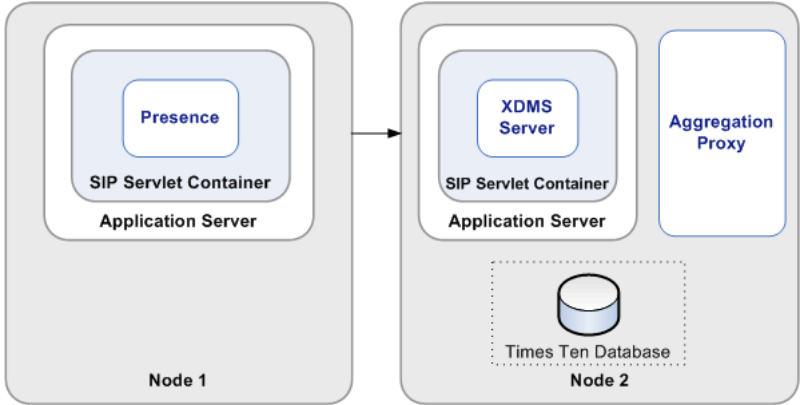
Two Node Presence Server

The Presence Server topology includes hardware and software components described in Table 3–4.

Table 3–4   *Presence Server Topology Hardware and Software Requirements*

Hardware	Software	Installation Type[1]
One computer with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ Presence Server	Typical installation
One computer with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ XDMS Server ■ Aggregation Proxy ■ Oracle TimesTen database (required for the Aggregation Proxy)	Typical installation

[1] Refer to the *OCMS Installation Guide* for more information.

XDMS Server

Manual post-installation configuration is required for the XDMS Server, involving configuring Presence as an XDMS Server. For more information, refer to the *OCMS Installation Guide*.

> **See also:**   See "Topology Components" for a description of the components used in this topology.

Deploying OCMS as an Instant Messaging Platform

The OCMS Instant Messaging topology is a highly available, six node topology that enables instant messaging client applications. The IM topology comprises a four node SIP network topology and a two node Presence Server topology, with the addition of the Application Router. The Application Router routes SIP requests to either the Proxy Registrar or the Presence Server, enabling registering and retrieving user contact and location data, as well as handling all aspects of Presence publication and notification. The Aggregation Proxy on either SIP Application Server node is used to authenticate subscriber access to presence documents stored on the XDMS Server. The IM topology provides the server-side functionality behind instant messaging client applications.

Figure 3–4   Instant Messaging Topology
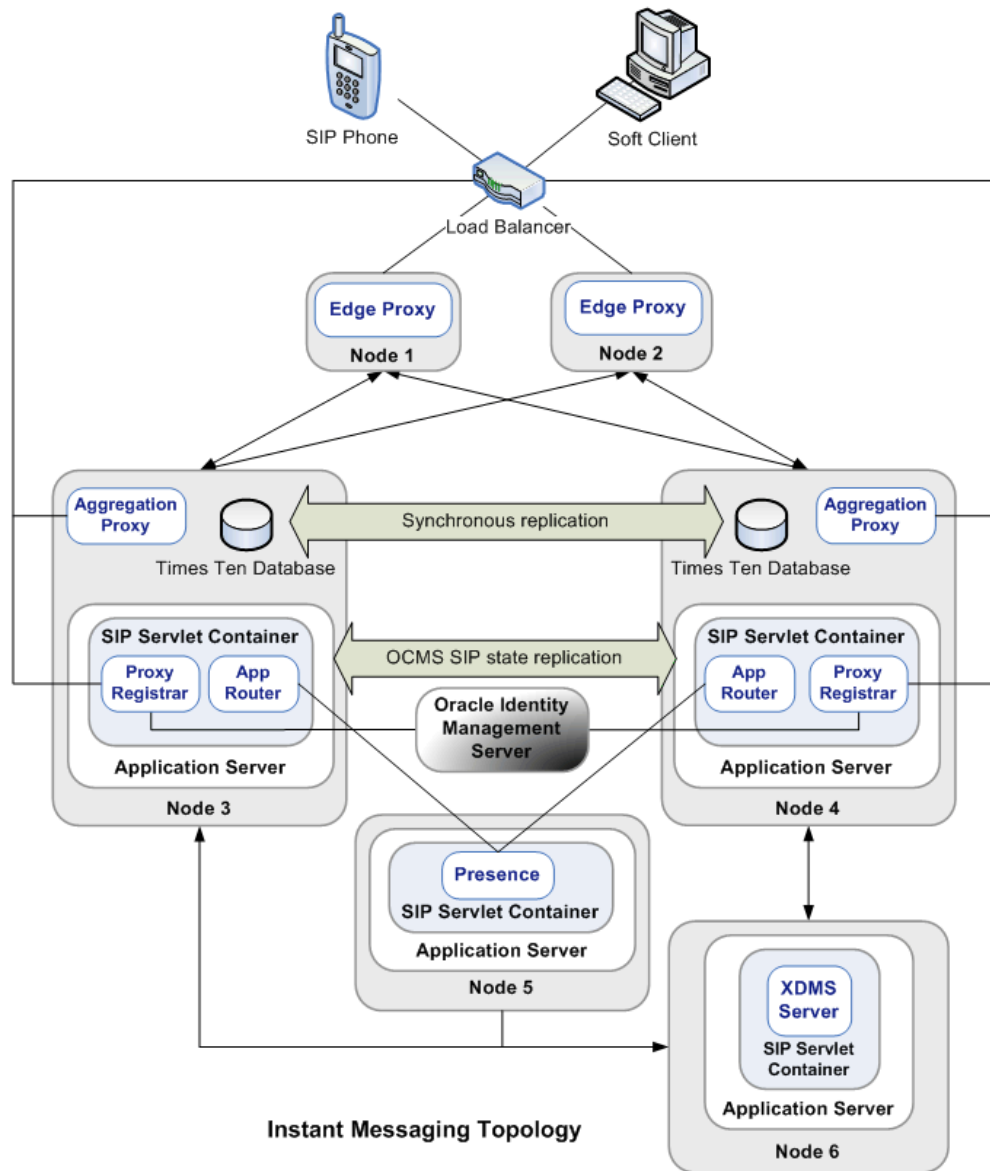
The Instant Messaging topology includes hardware and software components described in Table 3–5.

Table 3–5   Instant Messaging Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
Load balancer		
Two computers with at least 4 GB of RAM and a dual 2.8 Ghz CPU	Edge Proxy	Custom installation
Two computers with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ Oracle TimesTen Database ■ Aggregation Proxy ■ Application Router ■ Proxy Registrar	Typical installation

Table 3–5 (Cont.) Instant Messaging Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
One computer with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ Presence Server	Typical installation
One computer with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ XDMS Server	Typical installation

[1] Refer to the *OCMS Installation Guide* for more information.

For more information about scaling this hybrid topology, see "Deploying OCMS as a Highly Available SIP Network" and "Deploying OCMS as a Presence Server".

> **See also:** "Topology Components" for a description of the components used in this topology.

Deploying an OCMS Testing Environment

An OCMS testing environment is deployed on one node, and therefore does not provide high availability. A single node OCMS topology is appropriate for testing, demos, and small enterprises. Such an environment includes:

■ One SIP Application Server node

Figure 3–5 Single Node Deployment
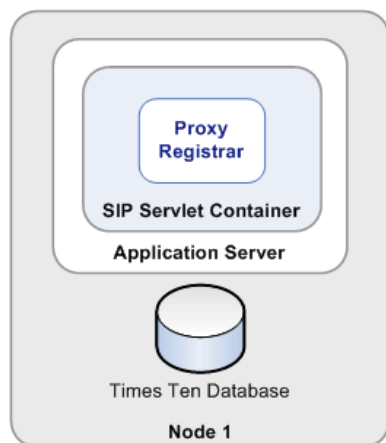
Single Node Proxy Registrar

Figure 3–6 Single Node Deployment with Presence
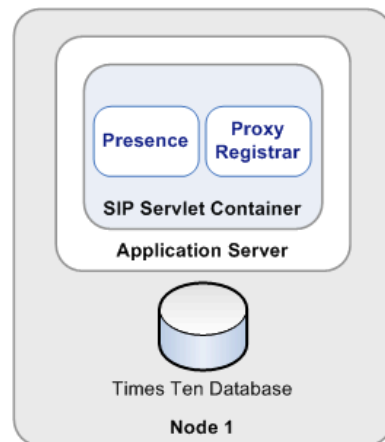
Single Node Proxy Registrar and
Presence

The single node topology includes hardware and software components described in
Table 3–6.

Table 3–6   Single Node Topology Hardware and Software Requirements

Hardware	Software	Installation Type[1]
One computer with at least 4 GB of RAM and a dual 2.8 Ghz CPU	■ OAS 10.1.3.2 ■ Oracle TimesTen database ■ Proxy Registrar	Typical installation
	If deployment includes Presence: ■ Presence application	Typical installation

[1]   Refer to the *OCMS Installation Guide* for more information.

See also: "Topology Components" for a description of the
components used in this topology.

Configuration Recommendations

The following is recommended for most topologies:

- **Use TCP as the transport protocol.** SIP clients use UDP or TCP to transport SIP
  messages. If there is a concern about exceeding UDP MTU size limits, TCP should
  be used. The preference for TCP can be enforced by adding NAPTR and SRV
  records to the DNS indicating that TCP is the preferred protocol. Make sure that
  clients connecting to OCMS fully support NAPTR and SRV records.

- **Run the OCMS SIP Server in the default Record-Route mode.** The best way for
  clients to handle NAT/FW in the network is to establish a TCP connection to the
  server upon registration, and reuse this connection for all incoming and outgoing
  SIP traffic. This requires the client to have an `outboundproxy` setting that points
  to the Proxy Registrar, and that the Proxy Registrar is configured to use
  `record-route`.

4

Configuring the SIP Servlet Container

This chapter discusses configuring the SIP servlet container in the following sections:

- "Overview"
- "Starting the SIP Servlet Container"
- "Accessing the System MBeans in the MBean Browser"
- "Configuring the SIP Servlet Container"
- "Configuring the Application Router"

Overview

Oracle Application Sever Control enables you to manage the SIP Servlet Container as an MBean (a Managed Bean).

OC4J provides the Oracle Enterprise Manager 10*g* Application Server Control Console, a JMX-compliant, Web-based user interface for deploying, configuring and monitoring applications within OC4J, as well as managing the OC4J server instance and the Web services used by your applications. You can also use the MBean Browser to manage other OCMS components (such as logging, and so on). See Chapter 9, "Managing the SIP Server" for more information.

The SIP servlet container system MBean may be accessed from the Administration tab of the Application Server Control Console. This MBean exposes attributes and operations that enable configuration of the SIP servlet container.

The main steps to configuring the SIP servlet container are as follows:

- Starting the SIP Servlet Container
- Accessing the System MBeans in the MBean Browser
- Configuring the SIP Servlet Container
- Configuring the Application Router

Starting the SIP Servlet Container

The SIP Servlet container runs on:

- UNIX/Linux—Starting the SIP Servlet Container on Linux or UNIX
- Windows—Starting the SIP Servlet Container on Windows

Starting the SIP Servlet Container on Linux or UNIX

To start or stop the SIP servlet container on Linux or UNIX:

- Navigate to the following directory:

  ```
  $ORACLE_HOME/sdp/bin
  ```

- At the command line, execute the following command:

  ```
  startocms
  ```
  Or

  ```
  stopocms
  ```

Starting the SIP Servlet Container on Windows

To start or stop the SIP servlet container on Windows:

- Select **Start > Programs > Oracle Communication and Mobility Server > Start OCMS** or **Stop OCMS**.

Accessing the System MBeans in the MBean Browser

To manipulate the system MBeans in the Application Server Control Console MBean Browser:

1. Type the following URL in a Web browser:

   ```
   http://<hostname>:<port>/em
   ```

   where `hostname` is the address of the application server running the SIP servlet container and `port` is the port number.

   To check the port number of the application server instance, execute the following command:

   ```
   ORACLE_HOME/opmn/bin/opmnctl status -l
   ```

   By default, the port number for a single node deployment of OCMS is 8888.

2. In the Application Server Control home, click **Administration** and scroll to the bottom of the page.

3. Under JMX, click the icon for **System MBean Browser**.

4. Expand the **SipContainer** folder and select the SipServletContainer attribute.

5. In the right pane, click **Attributes** to view current configurations or **Operations** to set parameters.

See the online Help provided with Application Server Control Console for detailed instructions on using this interface.

> **Note:** Certain changes made through the MBean browser are lost after the system is restarted. Configuration changes to the SIP container made through the MBean browser are persisted to the file system and will survive a server restart.

Configuring the SIP Servlet Container

The SIP servlet container includes a number of configurable parameters, some of which should be configured prior to deploying SIP applications. You can configure the SIP servlet container parameters by accessing the System MBeans in the AS Control Console.

The following parameters should be configured prior to deploying applications:

- Supported SIP Traffic Transport Protocols—Configure the communication protocols that the SIP servlet container supports, such as TCP, UDP, and so on.

- Domains and Realms—Configure the domains you want to allow to connect to the SIP servlet container.

- Contact—Configure the public listen address to embed in the headers of outgoing SIP messages.

- DefaultApplications—Configure the SIP applications to which SIP messages are routed by default.

> **Note:** By default, the SIP servlet container IP address and SIP port, as well as the DNS caching server are configured during OCMS installation. Refer to the *OCMS Installation Guide* for more information.

Supported SIP Traffic Transport Protocols

The SIP servlet supports the following transport protocols for sending SIP messages:

- TCP (recommended)

- UDP

To enable transport protocols:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Attributes tab, select **true** or **false** from the drop-down list to enable or disable any of the following:

   - **UseTCP**

   - **UseUDP**

3. Click **Apply**.

To view enabled transport protocols:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Attributes tab, scroll down to the bottom of the page to view the enabled protocols (UseTCP, UseUDP).

Domains and Realms

The DomainsAndRealms attribute maps SIP domains to Java authentication realms. This mapping is used to dynamically challenge incoming SIP requests with the relevant authentication realm during SIP authentication. For example, when responding to a challenge issued by the SIP container, a domain or realm mapping for company.com requires the SIP URI user@company.com in order to successfully authenticate.

To configure a domain:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Operations tab, click **addDomain** and fill in the following parameters:

   - **domain**—The domain for which you want to allow access to the SIP servlet, such as `voip.com`.

   - **realm**—The name of the authentication realm with which the domain is associated. For example, `voip`.

3. Click **Invoke Operation**.

To remove a domain from the list of permitted domains:

- To remove a domain, click **removeDomain**, enter the domain name in the space provided and click Invoke Operation.

To view configured domains:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Attributes tab, click **DomainsAndRealms** to display a comma-separated list of configured hosted domains.

Contact

The Contact attribute includes contact information that a User Agent Client embeds in outgoing SIP requests, such as the host, port, and transport protocol to be used. This gives the SIP Server an "address" to which it can respond. Using the Contact attribute requires running a SIP Servlet that acts as a User Agent Client.

To configure the contact attribute:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Operations tab, click **setContactAsString**.

   Alternatively, click **setDistributableContactAsString** to set the Contact attribute for distributable applications used in a high availability environment.

3. Enter the domain name, port number, and transport protocol separated by commas, for example, `my.host.com,5060,tcp`.

4. Click **Invoke Operation**.

To view the configured Contact attribute:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Attributes tab, click **Contact**.

DefaultApplications

The DefaultApplications attribute represents a comma-separated list of applications that are invoked if no application matches a request. By default, the Application Router is configured as the default application.

To configure the default applications:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Operations tab, click **setDefaultApplicationsAsString** and enter a comma separated list of default applications in the space provided.

3. Click **Invoke Operation**.

To view the default applications:

1. Access the SipServletContainer MBean (see "Accessing the System MBeans in the MBean Browser").

2. In the Attributes tab, scroll down to the **DefaultApplications** attribute.

Configuring the Application Router

It is possible to configure the Application Router in standard or incremental modes. For more information on the Application Router and its modes, see "Application Router".

> **Note:** Before configuring the Application Router, make sure the `DefaultApplications` attribute is set to the Application Router.

To configure the Application Router:

1. Access the Oracle Enterprise Manager 10g Application Server Control.

2. Click the **Applications** tab.

3. Under **subscriberdataservices**, click **ocmsrouteloaderear**.

4. Click the **Administration** tab.

5. Under JMX, click **Application Defined MBeans**.

6. Click **Administration > System MBean Browser**.

   An MBean displays for each category of SIP traffic as follows:

   - `RouteLoader_Invite`

   - `RouteLoader_Message`

   - `RouteLoader_Publish`

   - `RouteLoader_Register`

   - `RouteLoader_Subscribe`

   You must configure the Application Router behavior for each category of SIP message.

   To configure the Application Router to run in standard mode, see Configuring the Application Router in Standard Mode.

   To configure the Application Router to run in incremental mode, see Configuring the Application Router in Incremental Mode.

Configuring the Application Router in Standard Mode

To set the Application Router to operate in standard mode, you must configure the following parameters:

- **IncrementalMode**—Set to false in order to run the Application Router in standard mode.

- **RecordRoute**—Set the RecordRoute parameter to true so that the SIP Servlet container records the route of the SIP messages.

- **SipUriList**—Configure the URI, port number, transport method (TCP or UDP), and application ID of each application to which you want to route SIP messages.

To configure the Application Router in standard mode:

1. Access the Application Router Mbeans, as described in Configuring the Application Router.

2. Configure the following for each Application Router Mbean:

   a. In the Operations tab, click **setRecordRoute** and select true. Click **Invoke Operation**.

   b. In the Operations tab, click **setSipUriList**. In the space provided, enter the URI of each application you want to include in the route header. By default, the Proxy Registrar is listed.

      For each application, enter the URI and port number, the transport method (TCP or UDP), and the name of the application. Enter the information in the following format:

      ```
      sip:111.11.111.1:5060;transport=TCP;lr;appId=proxyregistrar
      ```

      The order of comma separated applications determines the order in which SIP messages are routed to these applications.

   c. In the Operations tab, click **setIncrementalMode**. Set its value to **false** and click **Invoke Operation**.

Configuring the Application Router in Incremental Mode

To set the Application Router to operate in incremental mode, you must configure the following parameters:

- **IncrementalMode**—Set to true in order to run the Application Router in incremental mode.

- **SipUriList**—Configure the URI, port number, transport method (TCP or UDP), and application ID of each application to which you want to route SIP messages.

- **RecordRoute**—Set the RecordRoute parameter to true so that the SIP Servlet container records the route of the SIP messages.

- **RouteLoaderUri**—Configure the URI of the return route to the Application Router. This URI must be the same as that of the SIP Servlet container. In general, this URI should be set to the URI of the Proxy/Registrar, as the SIP Servlet container recognizes the URI of the Proxy/Registrar as its own

To configure the Application Router in incremental mode:

1. Access the Application Router Mbeans, as described in Configuring the Application Router.

2. Configure the following for each Application Router Mbean:

   a. In the Operations tab, click **setRecordRoute** and select true. Click **Invoke Operation**.

b. In the Operations tab, click **setSipUriList**. In the space provided, enter the URI of each application you want to include in the route header. By default, the Proxy Registrar is listed.

For each application, enter the URI and port number, the transport method (TCP or UDP), and the name of the application. Enter the information in the following format:

```
sip:111.11.111.1:5060;transport=TCP;lr;appId=proxyregistrar
```

The order of comma separated applications determines the order in which SIP messages are routed to these applications.

c. In the Operations tab, click **setIncrementalMode**. Set its value to **true** and click **Invoke Operation**.

d. In the Operations tab, click **setRouteLoaderURI**. In the space provided, enter the URI of the Proxy/Registrar in the following format:

```
sip:152.68.199.9;lr;
```

5

Provisioning Users and Applications

This chapter describes using the Sapphire Shell (Sash) utility. This chapter includes the following sections:

- "Overview of Sash"

- "Launching Sash"

- "Using Sash"

- "Creating a User"

- "Provisioning the XDMS Using Sash"

- "Scripting with Sash"

- "Error Logging in Sash"

Overview of Sash

The Sapphire Shell (Sash) is a command-line utility to provision OCMS users to the TimesTen In-Memory 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 "Configuring Oracle Internet Directory as the User Repository" for information on using Oracle Internet Database (OID), the LDAP data store used by Oracle WebCenter Suite, as the user provisioning repository for an OCMS deployment.

Launching Sash

On Windows, you invoke Sash by selecting the *Launch Sash* option of *Oracle Communications and Mobility Server* (accessed through **Start** > **All Programs**). On Linux systems, the Sash launcher script (launch_sash.sh) is located in the same folder that contains the start and stop scripts for OCMS.

Figure 5–1   Launching Sash from the Windows Programs Menu
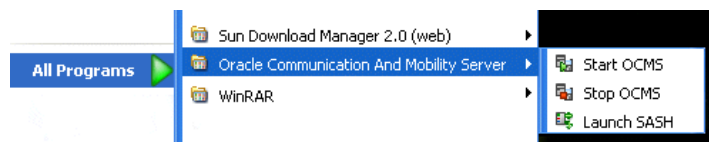
Launching Sash from the Command Line

You can also launch Sash using the batch or shell script files (`sash.bat` and `sash.sh`, respectively) which are located in OC4J OCMS installations at ORACLE_HOME/sdp/sash/sbin and on JBoss Application Server installations at jboss-4.0.5.GA/sdp/sash/sbin.

OCMS provides the following shortcut for launching Sash from the command line:

`launch_sash.sh` (UNIX)

`launch_sash.bat` (Windows)

This shortcut is located at ORACLE_HOME/sdp/bin on OC4J OCMS installations and at JBOSS_HOME/sdp/bin on JBoss Application Server installations.

Secured and Unsecured Connections to JBoss Application Server

JBoss Application Server supports both secured and unsecured installations. Unsecured installation do not require authentication credentials to connect to the JBoss MBean Server and JMX Console. In secure installations, users must provide user names and passwords to connect to the JBoss MBean Server and JMX Console. In the Sash command line, the `auth` flag denotes secured connections from Sash to JBoss and has the effect of prompting users for user names and passwords. The syntax is as follows:

```
#sash--jboss auth
```

In unsecured JBoss deployments, the Sash command line is invoked with the `noauth` flag, meaning that users do not have to provide authentication credentials. The syntax is as follows:

```
#sash--jboss noauth
```

The `auth` and `noauth` flags are determined by whether the JBoss instance has been configured in a secured mode. If you use the *Launch Sash* option of *Oracle Communications and Mobility Server*, then the `auth` and `noauth` flags are set appropriately for the environment based on information provided during the OCMS installation.

Connecting Sash to an External OCMS Instance

By default, Sash connects to the local instance of OCMS. If needed, you can override this default behavior and connect Sash to external instances of OC4J or to another instance of Oracle Application Server, or JBoss Application Server.

Connecting to an External Instance of OC4J

Sash connects to the OCMS server through RMI. Example 5–1 illustrates how to connect Sash to a server with the host IP address 10.0.0.234.

Example 5–1   Connecting Sash to OCMS

```
sash#--host 10.0.0.234
```

When you connect to OC4J, Sash prompts you for a username and a password. The user name is the same as that for OC4J administrator (oc4jadmin). The password is the same as the password associated with the OC4J administrator. Once you log in, the Sash command prompt (`sash`) appears. An error message displays if the login is unsuccessful.

Connecting Sash to an External Oracle Application Server Instance

To connect Sash to an external instance of the Oracle Application Server, use the `--ias` option and then add `-- port` followed by the port number for OPMN (Oracle Process and Management and Notification) server. Example 5–2 illustrates how to connect to an external instance of the Oracle Application Server.

Example 5–2   Connecting to an External Instance of the Oracle Application Server

```
sash#--ias --port 6003
```

When you connect to the server, enter the administrator user name and password.

> **Tip:**   The OPMN request port is configured in the `opmn.xml` configuration file under ORACLE_HOME/opmn/conf/opmn.xml.

Connecting Sash to an External JBoss Application Server Instance

To connect Sash to an external instance of the JBoss Application Server, use the `--jboss` option and enter host number. For example, to connect to an application server with the host IP address of 10.0.0.234, enter the following:

```
#sash--jboss -- host 10.0.0.234
```

When you connect to the server, enter the administrator user name and password (if prompted).

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.

Viewing Available Commands

Entering `help` displays a list of all available commands in the server (described in Table 5–1). The list of commands varies depending on the components deployed to the server.

Table 5–1   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 5–1   (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. OCMS 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 5–1 (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".

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 5–3).

Example 5–3   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 5–3 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.

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 5–1 to provision a user account to the TimesTen In-Memory 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.

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 5–4) by creating an OCMS user known as *alice* using the commands described in Table 5–1.

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 5–1 to set the status of the account. To create an active account for alice, enter the following:

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

> **Note:** OCMS Version 10.1.3.2 requires that the `uid` be in lower-case. Oracle Communicator users provisioned using OCMS Version 10.1.3.2 must also enter their account names in lower case during login. OCMS Version 10.1.3.3 supports 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 10.1.3.3 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
   ```

   `realm` and `password` are only needed to store authentication information in the TimesTen In-Memory database (accessed through the OCMS LoginModule). 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 6, "OCMS Security".

   > **Note:** You must also configure `realms` using the SIP Servlet Container MBean before you use Sash to add authorization credentials to a user. For more information, see "Configuring the SIP Servlet Container MBeans".

Example 5–4   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".

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 Subscriber Data Services application. For information on accessing application-defined MBeans, see "Accessing the MBeans for a Selected SIP 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 (Figure 5–2).

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**.

Figure 5–2   Creating a User with the Command Service MBean

See "CommandService" for more information on executing Sash commands through Application Server Control.

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 OCMS 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:** The `realm` and `password` attributes are only needed for applications configured to authenticate users against the TimesTen In-Memory database through the OCMS Login Module. 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"
> ```
>
> See Chapter 6, "OCMS Security" for more information on the login modules.

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 `publicIds` with a single `privateId`, must be created using multiple Sash commands as illustrated in Example 5–4.

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
```

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.

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 (Figure 5–3). 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)".

Figure 5–3   Using the CommandService MBean for XCAP Account Management

Provisioning XDMS User Accounts from the Sash Prompt

To use the XDMS commands to provision users and application usages from the Sash prompt, you must first connect to an XDMS-consuming application such as Presence by entering `sash -a <application name>` from the command prompt (Figure 5–4).

> **Note:**   You connect to the XDMS-consuming application through a command prompt, such as the Windows command shell (`Cmd.exe`). You cannot connect to these applications directly from the Sash prompt.

For example, to connect to the Presence application:

1. From the command prompt, navigate to the `\sbin` directory that contains the Sash executable. This file is located at Oracle_HOME\sdp\sash\sbin.

2. Enter the name of the Presence application using `sash -a presenceapplication`. For example, enter the following:

    ```
    c:\product\10.1.3.2\ocms\sdp\sash\sbin>sash -a presenceapplication
    ```

3. When prompted, login to Sash using your OC4J administrator name and password.

4. From the Sash command prompt, enter an XDMS command, such as `xcap user list`.

Figure 5–4   Connecting to the Presence Application

Using xcap Commands

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

Provisioning XDMS User Accounts

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

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> applicationUsage=<string>
```

> **Caution:**   Do not use the `add` command if the XDMS is configured to automatically create users.

Removing an XDMS User

The `xcap user delete` command removes an XDMS user with the given user name 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. When the server is configured to automatically create a user, the `delete` command removes all existing documents.

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>
```

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.

Provisioning Application Usage

The commands for provisioning of XDMS application usage are in the `appusage` group (`xcap appusage`).

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
```

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 5–5 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 ocms_users.txt` from the Sash prompt:

Example 5–5   Creating Users from a Text File (ocms_users.txt)

```
identity add publicId=sip:alice@doc.oracle.com privateId=alice role=user
password=1234 realm=doc.oracle.com
identity add publicId=sip:bob@doc.oracle.com privateId=bob role=user password=1234
realm=doc.oracle.com
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
identity add privateId=gretchen publicId=sip:gretchen@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=hans publicId=sip:hans@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=imogen publicId=sip:imogen@doc.oracle.com role=user
password=1234 realm=doc.oracle.com
identity add privateId=jack publicId=sip:jack@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.

Error Logging in Sash

If a command encounters an error while executing, the error message is written to the *ERROR* log level, while success messages are written to the *INFO* log level.

By default, Sash uses two log4j appenders: one that listens on *INFO* log level, and one that listens on the *ERROR* (or greater) log level. The latter is configured to output the log level before any message. For example, you can locate an error message by searching the message for the [ERROR] string. The first appender is directed to stdout and the second is directed to stderr in the default configuration. The Sash logging output can be configured in the log4j.xml file located at ORACLE_ HOME/sdp/sash/conf directory. For more information on log4j, see Chapter 10, "Configuring the Logging System".

6

OCMS Security

This chapter describes how to set the security provider for an application. This chapter includes the following topics:

- "Overview of Security"
- "Configuring Applications to Use Login Modules"
- "Configuring Login Modules for the JBoss Application Server"
- "Security in SIP Servlets"
- "Authentication Using the P-Asserted Identity Header"
- "Authentication of Web Service Calls and XCAP Traffic"

Overview of Security

OCMS implements both basic (HTTP) authentication and digest authentication as described in RFCs 3261 and 2617. In OCMS, both SIP and HTTP applications can be configured to authenticate against the TimesTen In-Memory Database that stores users, user roles, and user data, a RADIUS authentication system, or the Oracle Internet Directory (OID).

The Subscriber Data Services application provides the infrastructure for authentication and authorization. This application, which is parent to applications requiring authentication, enables its child applications to access the authentication backend (TimesTen, RADIUS or Oracle Internet Directory) by means of native Enterprise JavaBeans (EJBs). Subscriber Data Services provides a framework that enables security but does not impose security constraints itself; this is done instead by its child applications, such as Proxy Registrar, which have authority constraints defined by the `<security-constraint>` element in `sip.xml`. For more information on configuring security in the deployment descriptor file, see "Security in SIP Servlets".

Each of the child applications of Subscriber Data Services must be configured to use a JAAS (Java Authentication and Authorization Service) login module for authentication and authorization against the configured user repository.

The OCMS JAAS-Compliant Login Modules

OCMS leverages the pluggable architecture of JAAS by providing the following modules that check user credentials stored in the TimesTen or RADIUS user repositories for SIP or HTTP applications:

- OCMS Login Module

The OCMS Login Module provides authentication and authorization against the users provisioned to the TimesTen In-Memory database. These users are provisioned using the CommandService MBean or through the Sapphire Shell (Sash) as described in Chapter 5, "Provisioning Users and Applications". The OCMS Login Module provides digest authentication for SIP applications and basic authentication for HTTP applications. For converged applications (applications containing both SIP servlets and HTTP), the OCMS Login Module provides basic authentication.

> **Note:** User provisioning depends upon the type of user repository and login module. While you can provision users to the TimesTen In-Memory database using Sash and then authenticate and authorize these users using the OCMS Login Module, you can also use Sash to provision users to the TimesTen In-Memory database while using an external database based on the RADIUS (Remote Authentication Dial-In User Service) protocol to store user passwords.

- RADIUS Login Module

  The RADIUS Login Module provides digest authentication for SIP applications as described in RFC 4590 and basic authentication HTTP applications for users provisioned to a RADIUS database.

  The RADIUS database only authenticates users. Unlike the TimesTen In-Memory database, RADIUS is not used to authorize users. Therefore, if you use the RADIUS Login Module, you must create users and assign the appropriate roles in the TimesTen In-Memory database that correspond to each user provisioned to the RADIUS database for role-based authorization. When deleting a user's authentication information from a RADIUS database, you must also delete that user's authorization information (for example, an account and role-related information) manually from the TimesTen In-Memory database. This prevents a future user provisioned with the same user name in RADIUS from inheriting the authorization and account-specific information of a user who had previously been deleted.

  > **Note:** Due to licensing restrictions, open source RADIUS is not packaged with OCMS during installation. Prior to configuring the RADIUS Login Module with OCMS, the JRadius client library must be manually downloaded and installed. The necessary JRadius client library can be downloaded from the Sourceforge project at:
  >
  > `http://jradius-client.sourceforge.net/`
  >
  > Once the JRadius client library has been downloaded, manually copy the `jradius-client.jar` file to `$ORACLE_HOME/j2ee/home/lib/ext` on Oracle Containers for J2EE (OC4J) deployments or to `$JBOSS_HOME/server/default/lib` on JBoss Application Server deployments. You must then restart the application server to deploy the RADIUS Login Module.

These pluggable JAAS login modules enable the SIP servlet container to perform authentication and authorization against external databases for User Agents sending SIP requests. The modules implement user authentication against the TimesTen or RADIUS user repositories by first invoking the JAAS `LoginModule` class and then by

authorizing a previously authenticated user by verifying that the appropriate access permissions have been granted.

> **Note:** Because these login modules authenticate users against external repositories, they are considered custom security providers in OC4J. See *Oracle Containers for J2EE Security Guide* for information on configuring a custom security provider within a J2EE application.

Application Type and Authentication Mode

The type of application (SIP, HTTP, or converged) dictates the authentication mode used by login module. As noted in Table 6–1, HTTP and converged applications are limited to basic authentication. In particular, this limitation applies to third-party HTTP applications that are deployed to OCMS and use the J2EE container to invoke the OCMS Login module.

> **Note:** Although the Aggregation Proxy is a Web application, it is not limited to basic authentication because it explicitly performs authentication using internal security APIs. As a result, the Aggregation Proxy's default security settings (which set the authentication method as digest) will work without further configuration.

Table 6–1 Authentication Based on Application Type

Application Types	Authentication Mode
SIP	Basic or Digest
HTTP and Converged	Basic Only

> **Note:** This chapter describes how to specify the login module used by an application and how to configure the OCMS and RADIUS login modules themselves. See "Configuring Oracle Internet Directory as the User Repository" for information on using Oracle Internet Directory (OID), the LDAP data store used by Oracle WebCenter Suite, as the user provisioning repository for an OCMS deployment.

Configuring Applications to Use Login Modules

You configure security for SIP applications by first defining the `<security-constraint>` element in the deployment descriptor file, `sip.xml`, setting the security provider (login module) appropriate to the authenticating user repository used by the application and then by configuring the login module itself. For more information on setting security in `sip.xml`, see "Security in SIP Servlets".

You can configure a login module for each application that you deploy. During the deployment process, you can configure the login module for SIP or HTTP applications using the *Security Provider* task in the *Deployment Settings* page of the Application Server Control deployment wizard. Once you deploy an application, you can examine or edit its login module's configuration From the *Edit Login Module* page, accessed from the *Security Providers* task in the *Administration* page (Figure 6–1).

Figure 6–1 Modifying a Login Module.

For more information, see *Deploying with Application Server Control Console* in *Oracle Containers for J2EE Deployment Guide*

Configuring Login Modules though system-jazn-data.xml and orion-application.xml

Alternatively, login modules can also be configured through the `<jazn-loginconfig>` settings either in the `system-jazn-data.xml` file or in the `orion-application.xml` file located in an application's EAR (Enterprise Archive) file.

Configuring Login Modules in system-jazn-data.xml

The `system-jazn-data.xml` file is the repository for login module configuration. The settings in this file are updated when you administer login modules through Oracle Application Server Control (Figure 6–1) or through the OracleAS JAAS Provider Admintool. For more information, see *Oracle Containers for J2EE Security Guide*.

Table 6–2 describes the options supported by the OCMS Login Module and the RADIUS Login Module that you configure in `system-jazn-data.xml`.

Table 6–2 Login Module Options

Option	Description
useUTF8	If set to `true`, then the login module supports user names and passwords encoded in the UTF-8 character set.
authMethod	For SIP applications, select either `Basic` or `Digest`. For HTTP applications, select `Basic`.

Table 6–2   (Cont.) Login Module Options

Option	Description
failurePeriod	The time, in seconds, that an account is locked. The value entered in `orion-application.xml` takes precedence over the value entered in the `DefaultLockDuration` attribute of the AA Service MBean. See also "SIP Servlet Container".
maxfailure	The number of failed login attempts before an account is blocked.

Declaring the OCMS Login Module in orion-application.xml

The `<jazn-loginconfig>` element in `orion-application.xml` defines the login modules used by an application. The configurations you set in this file are populated to `system-jazn-data.xml`. Example 6–1 and Example 6–2 illustrate configuring login modules for authentication against the TimesTen In-Memory database and a RADIUS authentication system, respectively.

Example 6–1 illustrates how the Proxy Registrar, a SIP application (defined in the `<application>` element), uses the OCMS Login Module to authenticate against the TimesTen In-Memory database. The OCMS Login Module class is declared within the `<login-module>` element. The `<control-flag>` element further describes the OCMS Login Module as required, meaning that the OCMS Login Module must succeed. The options for this module (described in Table 6–2) set the OCMS Login Module to authenticate user names and passwords encoded in UTF-8 character set. The `<failurePeriod>` element locks the account for 600 seconds after an unsuccessful login attempt. The `<maxFailure>` element sets the number of unsuccessful login attempts to five.

Example 6–1   Declaring the OCMS Login Module in orion-application.xml

```
<jazn-loginconfig>
        <application Key="name">
            <name>proxyregistrar</name>
            <login-modules>
                <login-module Key="class control-flag">
                    <class>
                        oracle.sdp.ocmsloginmodule.OCMSLoginModule
                    </class>
                    <control-flag>required</control-flag>
                    <options>
                        <option>
                            <name>useUTF8</name>
                            <value>true</value>
                        </option>
                        <option>
                            <name>authMethod</name>
                            <value>Digest</value>
                        </option>
                        <option>
                            <name>failurePeriod</name>
                            <value>600</value>
                        </option>
                        <option>
                            <name>maxFailure</name>
                            <value>5</value>
```

```
                                    </option>
                                </options>
                            </login-module>
                        </login-modules>
                    </application>
                </jazn-loginconfig>
```

Declaring the RADIUS Login Module in orion-application.xml

Like Example 6–1, Example 6–2 defines a login module for the Proxy Registrar application. This application, however, requires users to be authenticated against the RADIUS database. In addition to the options described in Table 6–2, the RADIUS Login Module also supports the options described in Table 6–3.

Table 6–3   RADIUS Login Module Options

Option	Description
hostname	The host name or IP address of the remote RADIUS server.
authPort	The destination port for authentication requests.
acctPort	The destination port for the accounting server.
sharedSecret	A String known only to the RADIUS server and the RADIUS client.

Example 6–2   Declaring the Radius Login Module in orion-application.xml

```
<jazn-loginconfig>
    <application Key="name">
        <name>proxyregistrar</name>
        <login-modules>
            <login-module Key="class control-flag">
                <class>
                    oracle.sdp.radiusloginmodule.RadiusLoginModule
                </class>
                <control-flag>required</control-flag>
                <options>
                    <option>
                        <name>useUTF8</name>
                        <value>true</value>
                    </option>
                    <option>
                        <name>authMethod</name>
                        <value>Digest</value>
                    </option>
                    <option>
                        <name>failurePeriod</name>
                        <value>600</value>
                    </option>
                    <option>
                        <name>maxFailure</name>
                        <value>5</value>
                    </option>
                    <option>
                        <name>hostName</name>
                        <value>127.0.0.1</value>
                    </option>
                    <option>
                        <name>authPort</name>
```

```
                                          <value>1812</value>
                                      </option>
                                      <option>
                                          <name>acctPort</name>
                                          <value>1813</value>
                                      </option>
                                      <option>
                                          <name>sharedSecret</name>
                                          <value>secret</value>
                                      </option>
                                      <option>
                                          <name>radiusClientClass</name>

<value>oracle.sdp.radiusloginmodule.JRadiusClient</value>
                                      </option>
                                  </options>
                              </login-module>
                          </login-modules>
                      </application>
                  </jazn-loginconfig>
```
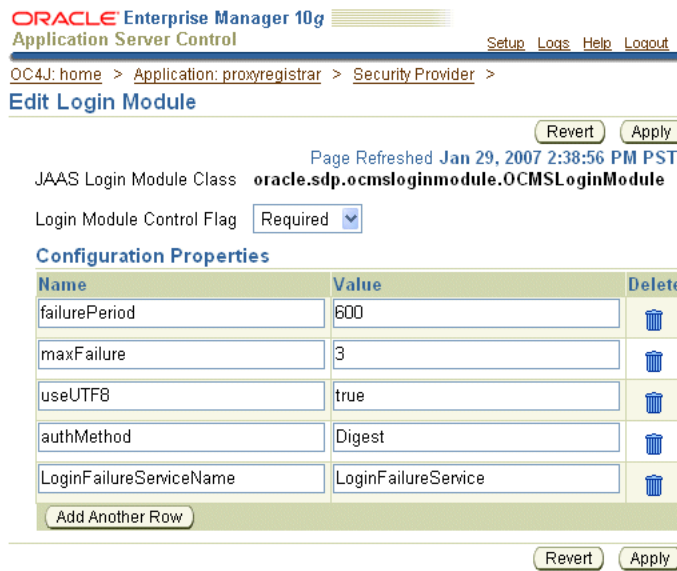
Configuring Login Modules for the JBoss Application Server

You must configure a security domain for each application that you deploy through the JBoss Application Server. You reference this domain for each application in the `<security-domain>` element (illustrated in Example 6–3) in `jboss-app.xml`.

Example 6–3   Defining a Security Domain

```
<security-domain>java:/jaas/OCMSSecurity</security-domain>
```

For SIP applications, define the value for `<security-domain>` as `OCMSSecurity` (for digest authentication) or `OCMSSecurityBasic` (for basic authentication). For HTTP applications, enter `OCMSSecurityBasic` as the value for `<security-domain>`.

Security in SIP Servlets

OCMS supports declarative and programmatic security for SIP servlets as described in the SIP Servlet API.

Declarative Security

The SIP Servlet API describes declarative security as expressing an application's security structure that includes roles, access control, and authentication requirements in a form that is external to the application[1]. The deployment descriptor, `sip.xml`, is the vehicle for declarative security. Developers define how security should be effected in a deployed application by defining the `<security-constraint>` element. This element, which is described in detail in *Oracle Communication and Mobility Server Developer's Guide*, includes the following child elements (described in Table 6–4).

In OCMS, you first configure security for SIP applications by defining these elements. You can also set the login module used by the application in the `system-jazn-data.xml` file or in the `orion-application.xml` file as described in "Configuring Applications to Use Login Modules".

---

[1] SIP Servlet API, Version 1.0

> **Note:** You can prevent a SIP application from performing authentication by removing the constraints defined in `sip.xml` and then by redeploying the application.

Table 6–4 Child Elements of the <security-constraint> Element

Element	Description
`<proxy-authentication>`	If this element is present in `SIP.xml`, the container must challenge the user agent with a 407 (*Proxy Authentication Required*) response status code when authenticating an incoming request or return a 401 response (*Unauthorized*).
`<resource-collection>`	A set of servlets and SIP methods. This element describes the servlet that requires authentication and the SIP methods used for authentication.
`<auth-constraint>`	Indicates the user roles that are permitted access to this resource collection.
`<role-name>`	The name of a security role.

Programmatic Security

Programmatic security describes the security model from inside a servlet using the `SipServletMessage` methods `getRemoteUser`, `isUserInRole`, and `getUserPrincipal`.

Authentication Using the P-Asserted Identity Header

The SIP Servlet API 1.0 states that in addition to basic and digest authentication, a User Agent authenticates users through the P-Asserted Identity, a SIP header field that conveys the identity of an authenticated user between the nodes of a trusted domain. As described in RFC 3325, a proxy within a trusted domain can receive messages from both trusted and non-trusted nodes alike. In the case of the latter, the proxy authenticates the originator of the message using digest authentication. The proxy then creates the P-Identity Asserted header field from the identity that it derived from authentication and places this field into the message header which it passes to other entities. For example, an inbound proxy server authenticates a user and then inserts the P-Asserted Identity header field into the received SIP message. By inserting the P-Asserted Identity header field, other servers within the trusted domain (such as the Presence Server) do not have to perform authentication again.

OCMS supports the trusted domain identity assertion described in RFC 3325 through the *SipservletCommandInterceptors* attribute of its SIP Servlet Container Mbean.

Authentication of Web Service Calls and XCAP Traffic

The Aggregation Proxy authenticates any XCAP traffic and Web Service calls (which are conducted through HTTP, not SIP) by inserting one of the following headers that are set through the Aggregation Proxy Mbean:

- X_3GPP_ASSERTED_IDENTITY (the default)

- X_3GPP_INTENDED_IDENTITY

- X_XCAP_ASSERTED_IDENTITY

The XCAP traffic and Web Service calls are then proxied to their respective servers. See also "Securing the XDMS with the Aggregation Proxy". For more information on Web Services, see *Oracle Communication and Mobility Server Developer's Guide*.

7

Packaging and Deploying Applications

This chapter, through the following sections, describes packaging and deploying SIP servlet applications to application servers:

- "Overview of SIP Servlet Applications"
- "Deploying SIP Applications"

Overview of SIP Servlet Applications

A SIP application can be comprised of servlets, class files, static resources and content, along with descriptive meta information which unifies these elements. As specified in JSR-116, a SIP servlet application is a structured hierarchy of directories. For converged applications (those comprised of both HTTP and SIP), the root of the hierarchy serves as the document root for files published from a Web server.[1] Within the hierarchy of the SIP servlet application, the WEB-INF directory stores the directories containing the `sip.xml` deployment descriptor file (`/WEB-INF/sip.xml`), the utility classes available to the application loader class(`/WEB-INF/classes`), and the directories containing the JAR files, servlets, beans, and utility classes useful to the Web application (`/WEB-INF/lib`).

The Deployment Descriptor File

The SIP application's deployment descriptor file, `sip.xml`, is comprised of the following elements:

```
<sip-app>
   <context-param>...</context param>
   <session-config>...</session-config>
   <servlet>...</servlet>
   <servlet mappings>...</servlet mappings>
   <listener>...</listener>
   <security-constraint>...</security-constraint>
</sip-app>
```

The application's common parameters are set in within the `<context-param>` element. The `<session-config>` element defines the application sessions. The `<servlet>` element defines the servlet for the container through its `<servlet-name>` and `<servlet-class>` child elements. The `<servlet mappings>` element defines how the application's servlets respond to requests. The application's life cycle listener classes and error handling are defined within the `<listener>` element. Security is declared for each servlet using the

---

[1] SIP Servlet API, Version 1.0

<security-constraint> element. Refer to *Oracle Communication and Mobility Server Developer's Guide* for a full description of the sip.xml file's elements.

Development to Deployment

The cycle from development to deployment of a SIP servlet application is as follows:

- Creating a SIP servlet by extending javax.servlet.sip.SipServlet and then by overriding the required methods for a particular service.

- Defining the SIP servlet application's initialization parameters (servlet definitions) and invocation rules (servlet mappings).

- Creating the deployment descriptor file (sip.xml).

- Building and packaging the application files and the sip.xml file into a Servlet Archive format file (SAR file).

> **Note:** SIP applications deployed through JBoss AS must be called SIP Servlet Archive files (.ssr) rather than Servlet Archive (.sar) files. Although the formats of .ssr and .sar files are identical, The JBoss Application Server cannot dispatch files with the .sar extension. Because of this limitation, you must use the .ssr extension when deploying an application using JBoss Application Server.

- Packaging the SAR (or SSR) file into an Enterprise Archive (EAR) file.

- Deploying the EAR file to OC4J (or JBoss Application Server).

Once the SIP servlet application has been successfully deployed and started on OC4J, view the log files and test it using a softphone client such as the Oracle Communicator client. Refer to *Oracle Communication and Mobility Server Developer's Guide* for more information on developing SIP servlets using OCMS SCE and the Eclipse SDK.

Deploying SIP Applications

OCMS accepts Enterprise Archive (EAR) files and Web Application Archive (WAR), but not SAR (or SSR) files. Therefore, you must package the SAR (or SSR) as an EAR file to enable the deployment of the SIP application to OC4J.

An EAR file can contain SAR files, JAR files, Web Application Archive (WAR) and EJB modules as follows:

```
J2EEAppName.ear
   META-INF/
   application.xml
   orion-application.xml (optional)
WebModuleName.war
   static HTML files, such as index.html
   JSP pages
   images
   WEB-INF/
      web.xml (Standard J2EE descriptor)
      orion-web.xml (optional OC4J Web descriptor)
      classes/
         servlet classes, according to package
      lib/
         JAR files for dependency classes
SIPApplicationName.sar
   WEB-INF/
```

```
sip.xml (deployment descriptor)
web.xml (for converged applications)
classes/
    servlet classes, according to package
lib/*.JAR
    JAR files for dependency classes
```

For more information on deployment and EAR application structure, see *Oracle Containers for J2EE Developer's Guide*.

Deploying the SIP Application Using Oracle Application Server Control

Application Server Control provides a JSR 88-based deployment wizard, accessed by clicking the **Deploy** button on the *Applications* page. This wizard enables deployment and redeployment of J2EE applications and includes both task-oriented deployment plan editors for assigning or mapping the common deployment descriptors at deployment time as well as a generic deployment plan editor that enables you to access all deployment descriptors for advanced configuration. For information on undeploying and redeploying applications, see "Deploying, Undeploying, and Redeploying SIP Servlet Applications with Application Server Control".

> **Note:** Although you can change an deployment plan using Application Server Control, you cannot use Application Server Control to alter the `sip.xml` deployment descriptor file. For information on deployment plans, see *Oracle Containers for J2EE Deployment Guide*.

SIP applications deployed beneath the Subscriber Data Services application inherit security infrastructure and authentication-related EJBs (Enterprise Java Beans). This infrastructure is required to support authentication against the OCMS JAAS Security providers. For more information, see Chapter 6, "OCMS Security".

> **Note:** When an application has been undeployed, its MBeans are also undeployed.

Once an application has been deployed to the OC4J SIP Server instance, you can start or stop it using the `admin_client.jar` utility by executing the following command:

```
java -jar admin_client.jar uri adminId adminPwd -start|-stop appName
```

Figure 7–1 Deploying an Application

Deploying the SIP Application Using the admin_client.jar Command-Line Utility

You can deploy an EAR file from the command line using the `admin_client.jar` utility as follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file
path/filename-deploymentName appName [-bindAllWebApps [webSiteName]]
[-targetPath path] [-parent appName] [-deploymentDirectory path]
-enableIIOP [-iiopClientJar path/filename][-deploymentPlan path/filename]
```

For more information undeploying and redeploying applications using the `admin_client.jar` utility, see "Deploying, Undeploying, and Redeploying an Application Using the admin_client.jar Utility". See also *Oracle Containers for J2EE Configuration and Administration Guide* and *Oracle Containers for J2EE Deployment Guide*.

> **Note:** The `admin_client.jar` utility is installed by default in ORACLE_HOME/j2ee/home in an OC4J instance. OC4J must be started before this utility can be used.

8

Configuring High Availability

This chapter discusses configuring high availability through the following sections:

- "About Configuring High Availability"
- "Setting Up a Highly Available Cluster of OCMS Nodes"
- "Configuring the OCMS SIP Containers for High Availability"
- "Configuring the Edge Proxy Nodes for High Availability"
- "Configuring Highly Available SIP Servlet Applications"
- "Configuring the Proxy Registrar for High Availability"

> **Note:** Currently, OCMS supports high availability using Oracle Application Server only.

About Configuring High Availability

OCMS provides high availability through redundancy, application state replication, and clustering. Highly available OCMS topologies are active-active, meaning that any redundant nodes actively function in the context of the topology. This makes OCMS scalable as well.

Figure 8–1 Highly Available OCMS Topology
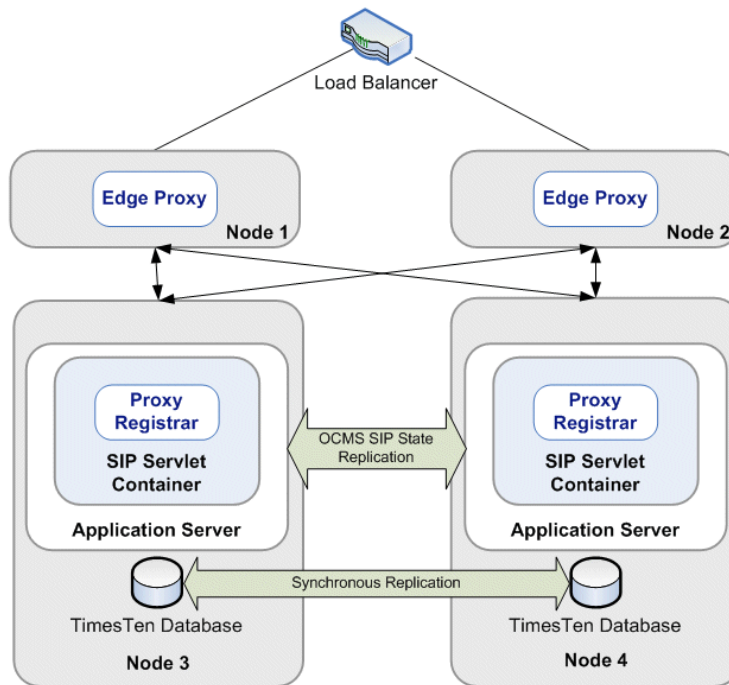
A highly available OCMS topology (Figure 8–1) provides the following:

- **Process death detection and automatic restart**—Processes may die unexpectedly due to configuration or software problems. A proper process monitoring and restart system monitors all system processes constantly and restarts them if necessary.

- **Clustering**—Clustering components of a system together allows the components to be viewed functionally as a single entity from the perspective of a client for runtime processing and manageability. A cluster is a set of processes running on single or multiple computers that share the same workload. There is a close correlation between clustering and redundancy. A cluster provides redundancy for a system.

- **Configuration management**—A clustered group of similar components often need to share common configuration. Proper configuration management ensures that components provide the same reply to the same incoming request, allows these components to synchronize their configurations, and provides highly available configuration management for less administration downtime.

- **State replication and routing**—For stateful applications, client state can be replicated to enable stateful failover of requests in the event that processes servicing these requests fail.

- **Server load balancing and failover**—When multiple instances of identical server components are available, client requests to these components can be load balanced to ensure that the instances have roughly the same workload. With a load balancing mechanism in place, the instances are redundant. If any of the instances fails, then requests to the failed instance can be sent to the surviving instances.

> **Note:** For more information about the architecture illustrated in Figure 8–1, refer to "Deploying OCMS as a Highly Available SIP Network".

Configuring a highly available OCMS environment involves the following main steps, depending on the OCMS topology you have chosen to deploy:

- Setting Up a Highly Available Cluster of OCMS Nodes—This involves associating each OCMS node with OPMN to form a manageable cluster.

- Configuring the OCMS SIP Containers for High Availability—Use the Application Server Control Console MBean browser to configure parameters affecting high availability in the SIP Servlet container.

- Configuring the Edge Proxy Nodes for High Availability—Optional. Required only if using Edge Proxy nodes in the OCMS topology.

- Configuring Highly Available SIP Servlet Applications—Edit the SIP Servlet application descriptor files to enable support for high availability. Configure state replication for each application.

- Configuring the Proxy Registrar for High Availability—Optional. Required only if using the Proxy Registrar in the OCMS topology.

Table 8–1   Additional Information

For more information on...	See:
OCMS deployment topologies	Chapter 3, "Deployment Topologies" in this guide
OCMS installation	*Oracle Communication and Mobility Server Installation Guide*
Operating systems supported by highly available OCMS clusters	*Oracle Communication and Mobility Server Certification Guide*
Configuring a highly available clustered Oracle Application Server environment	- The "Application Clustering" chapter in *Containers for J2EE Configuration and Administration Guide*. - The "Active-Active Topologies" chapter in *Oracle Application Server High Availability Guide*.

Setting Up a Highly Available Cluster of OCMS Nodes

> **Note:** If using UDP, place all servers on the same subnet or switch to avoid the defragmentation of large UDP packages.

Each OCMS node—including the Edge Proxy nodes—must be configured to support high availability.

Following are the main steps in setting up a cluster of OCMS servers:

1. Associating Nodes with OPMN—Oracle Process Manager and Notification Server provides a command-line interface for process control and monitoring for single or multiple Oracle Application Server components and instances. Using OPMN, you can start and stop each OCMS node and its sub-components.

2. Starting the Cluster—Starting the cluster with OPMN indicates that all OCMS nodes have been correctly associated with OPMN and are recognized as a cluster.

3. Verifying the Status of the Cluster—Using OPMN or Enterprise Manager, you can verify that each node in the cluster is up and running.

4. Stopping the Cluster—Having set up and verified the cluster of OCMS nodes, be sure to stop the cluster before configuring each SIP container for high availability (see "Configuring the OCMS SIP Containers for High Availability").

Associating Nodes with OPMN

Setting up a cluster of OCMS nodes requires associating the nodes with OPMN. There are three ways to do this:

- Configuring the cluster during Oracle Application Server installation. For more information, refer to the *Oracle Application Server Installation Guide*.

- Associating Nodes with OPMN Using the Dynamic Discovery Method

- Associating Nodes with OPMN Using the Discovery Server Method

> **See also:** For more information regarding configuring and managing clusters using OPMN, see *Oracle Process Manager and Notification Server Administrator's Guide*.

Associating Nodes with OPMN Using the Dynamic Discovery Method

In this method -- one that is recommended by Oracle -- you define the same multicast address and port for each Oracle Application Server instance in the cluster. An advantage in using this method is that you do not have to specify the name of each Oracle Application Server instance in the cluster. Instead, you can dynamically add or remove instances from the cluster by editing the multicast address and port.

1. For each Oracle Application Server instance that you want to group in the same cluster, run the following command:

```
opmnctl config topology update discover="*<multicastAddress>:<multicastPort>"
```

For example:

```
> ORACLE_HOME/opmn/bin/opmnctl config topology update
                      discover="*225.0.0.20:6200"
```

where:

- `multicastAddress` specifies the multicast address that you want to use for the cluster. The multicast address must be within the valid address range, which is 224.0.1.0 to 239.255.255.255. Note that the multicast address is preceded by an asterisk (*).

- `multicastPort` can be any unused port number.

Use the same multicast IP and port for all the instances.

2. On each Oracle Application Server instance where you ran the command in Step 1, run `opmnctl reload` so that OPMN reads the updated `opmn.xml` file.

```
> ORACLE_HOME/opmn/bin/opmnctl reload
```

Associating Nodes with OPMN Using the Discovery Server Method

Although Oracle recommends associating nodes with OPMN using the dynamic discovery method, you can also define a cluster by specifying the names of the nodes running the Oracle Application Server instances in the opmn.xml file for each instance. For example, to cluster four instances (*node1.example.com*, *node2.example.com*, *node3.example.com*, *node4.example.com*), associate these nodes with OPMN using the discovery server method as follows:

1. Run Oracle Application Server on all nodes.

2. Designate one instance as the discovery server, which maintains the topology for the cluster. (In this example, *node1.excample.com* acts as the discovery server for the cluster.)

3. In the opmn.xml file for all instances in the cluster, specify the node that is running the discovery server (*node1.example.com* in Example 8–1). As illustrated in Example 8–1, the opmn.xml file includes the <discover> element. The 6200 value specifies the port number on which the notification server listens. Use the remote port number designated in the <port> sub-element of the <notification-server> element.

Example 8–1   Designating an Instance as the Discovery Server

```
<?xml version="1.0" encoding="UTF-8"?>
<opmn xmlns="http://www.oracle.com/ias-instance">
   ...
   <notification-server interface="ipv4">
      <port local="6100" remote="6200" request="6003"/>
       ....
      <topology>
         <discover list="node1.example.com:6200"/>
      </topology>
      ...
   </notification-server>
   <process-manager>
   ...
   </process-manager>
</opmn>
```

4. On all server instances, run opmnctl reload so that OPMN loads the updated opmn.xml file:

```
ORACLE_HOME/opmn/bin/opmnctl reload
```

Starting the Cluster

To start the cluster using OPMN run the following command on each instance in the cluster:

```
cd ORACLE_HOME/opmn/bin/
opmnctl startall
```

Verifying the Status of the Cluster

To verify the status of the OCMS nodes in the cluster:

1. In a Web browser, enter the URI of Enterprise Manager running on any SIP container in the cluster:

```
http://<SIP container URI>:<port number>/em
```

2. Enter the administrator user name and password at the prompt.

Enterprise Manager displays the status of the cluster topology.

Stopping the Cluster

After verifying the status of the cluster, stop the nodes in the cluster using OPMN so that you can continue configuring the SIP containers (see "Configuring the OCMS SIP Containers for High Availability").

To stop OCMS, execute the following command on each node in the cluster:

```
cd ORACLE_HOME/opmn/bin/
opmnctl stopall
```

Configuring the OCMS SIP Containers for High Availability

In the Application Server Control Console MBean browser, configure the following parameters under the SIP Servlet Container MBean for each SIP Application Server node:

1. Configure the *EdgeProxy* parameter to point to the SIP URI of the Edge Proxy or to a third-party load balancer if more than one Edge Proxy is used.

   Use the following format:

   ```
   SIP:<Edge Proxy or Load Balancer IP address>:<port>;lr
   ```

2. Configure the *DistributableRecordRoute* parameter in the following format:

   ```
   SIP:<SIP Container IP address>:<port>
   ```

   Remove any appended transport methods (such as `transport=tcp`) to enable any type of transport to be used between the Edge Proxy and OCMS.

3. Configure the *RecordRoute* parameter using the following format:

   ```
   SIP:<SIP Container IP address>:<port>
   ```

   Remove any appended transport methods (such as `transport=tcp`) to enable any type of transport to be used between the Edge Proxy and OCMS.

Configuring the Edge Proxy Nodes for High Availability

> **Note:** In the load balancer, you must disable stickiness for UDP datagrams sent to the Edge Proxy servers. Refer to the load balancer documentation for more information on disabling stickiness when sending datagrams over UDP.

To configure the Edge Proxy nodes for high availability:

- Configure each OCMS node running an Edge Proxy for high availability as described in "Setting Up a Highly Available Cluster of OCMS Nodes" and "Configuring the OCMS SIP Containers for High Availability".

- Point the *edgeproxy* Mbean to one of the following:

  - The IP address of the Edge Proxy node.

- For more than one Edge Proxy node -- The virtual IP address or host name of the third-party load balancer or DNS server if clients connect using DNS lookup.

- Configure the interval at which SIP Container nodes ping the Edge Proxy nodes, as well as the number of missed ping intervals before the Edge Proxy nodes remove the unresponsive SIP Container from the routing table. These parameters enable the Edge Proxy node(s) to monitor the health of the SIP Container nodes.

For each Edge Proxy node in the topology, configure the following:

1. In the Application Server Control Console Mbean Browser, click the *edgeproxy* Mbean.

2. Configure the *RecordRoute* parameter to point to one of the following:

   - **For a single Edge Proxy without a load balancer**—Set the parameter to the IP address of the Edge Proxy node

   - **For more than one Edge Proxy with a load balancer or DNS server**—Set the parameter to the virtual IP address or host name of the third-party load balancer or DNS server (if clients connect using a DNS lookup)

3. Modify the `edgeproxy.xml` file (`sdp/edgeproxy/conf/edgeproxy.xml`, illustrated in Example 8–2) to include the `oc4j-ping` element:

   ```
   <oc4j-ping interval="1" allowed-miss-count="2"/>
   ```

   The `oc4j-ping` element configures the interval, in seconds, at which the Oracle Application Servers in the cluster ping the Edge Proxy. The `allowed-miss-count` attribute specifies the number of missed ping intervals allowed before the Edge Proxy removes an unresponsive Oracle Application Server from the routing table.

Example 8–2   edgeproxy.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<edge-proxy xmlns:xsi="http://www.oracle.com/sdp">
    <record-route sip-uri="sip:%IPADDRESS%:%SIPPORT%"/>
    <jmx-rmi-connector port="%EPRMIPORT%"/>
    <oc4j-ping interval="1" allowed-miss-count="2"/>
    <nat-traverse enabled="true"/>
    <sip-stack ip="%IPADDRESS%">
        <listening-point transport="tcp" port="%SIPPORT%" />
        <listening-point transport="udp" port="%SIPPORT%" />
    </sip-stack>
</edge-proxy>
```

4. In the `edgeproxy.xml` file, modify the `nat-traverse` element if necessary.

   - If the Edge Proxy enables SIP clients to traverse of NATs (Network Address Translators), then set the value to `true` (the default). The corresponding default value must be set in Oracle Communicator.

   - If not using NAT traversal, this attribute must be set to `false`. For more information disabling NAT traversal, see "Disabling NAT Traversal Enabled by the Edge Proxy".

5. Verify the status of the Edge Proxy node or nodes in the cluster by performing the following:

   - Starting the Cluster

■   Verifying the Status of the Cluster

For more information, refer to "Configuring OCMS in a Clustered Environment with Edge Proxy" in *Oracle Communication and Mobility Server Installation Guide*

The NAT Traversal Option Enabled for the Edge Proxy

NAT traversal enables access to SIP User Agents even when they are located behind firewalls or NATs. To support SIP clients residing behind firewalls or NATs, proxy servers use the Path extension header mechanism (described in RFC 3327), which ensures that SIP clients follow specific paths that enable the traversal of NATs and firewalls throughout the network. When you enable the NAT traversal function in the Edge Proxy, an OCMS cluster supports the Path extension header mechanism by inserting a Path header field into REGISTER requests.

Disabling NAT Traversal Enabled by the Edge Proxy

By default, NAT traversal is enabled in `edgproxy.xml` (`nat-traverse enabled=true`, as noted in Example 8–2). To disable this function:

1. If the Edge Proxy is running, stop it by entering the following command:

    `ORACLE_HOME/opmn/bin/opmnctl stopproc process-type=EdgeProxy`

2. Edit the `nat-traverse` element of `edgeproxy.xml` (located at `ORACLE_ HOME/sdp/edgeproxy/conf/edgeproxy.xml`) as follows:

    `<nat-traverse enabled="false"/>`

3. Start the Edge Proxy using the following command:

    `ORACLE_HOME/opmn/bin/opmnctl startproc process-type=EdgeProxy`

4. Repeat these steps for each Edge Proxy node in the OCMS cluster.

> **Caution:**   When NAT traversal is enabled, the Edge Proxy nodes insert their local IP addresses into the *RecordRoute* headers of SIP requests. Therefore, the Edge Proxy nodes must be globally routable. This may not be the case if the cluster has been configured according to the white paper, *Oracle Communication and Mobility Server in a High-Availability Environment Running with F5 BigIP* (available at the Oracle Technology Network).

Configuring Highly Available SIP Servlet Applications

This section describes how to configure high availability for SIP Servlet applications deployed to a cluster of OCMS nodes.

■   Enabling High Availability in SIP Servlet Applications—Prior to deployment, each application's descriptor files (`web.xml` and `sip.xml`) must be configured for high availability. The `orion-application.xml` file for each application must also be configured for high availability.

■   Configuring Application Session Data Replication—The session data of each SIP Servlet application can be replicated to other nodes in the cluster, in the event of node failure.

■   Configuring High Availability for a Deployed SIP Servlet Application—You can also configure high availability for an application that you have already deployed.

■ Disabling High Availability at the Application Level—You can remove support for high availability from your SIP Servlet applications.

■ Upgrading SIP Servlet Applications in OCMS—This section explains how to perform a rolling upgrade on deployed SIP Servlet applications.

> **Notes:**
>
> ■ When configuring high availability for SIP Servlet applications that depend upon the Proxy Registrar, you must also configure the Proxy Registrar for high availability. See "Configuring the Proxy Registrar for High Availability" for more information.
>
> ■ High availability is not currently supported for converged applications (meaning applications comprised of both SIP and HTTP servlets).

Enabling High Availability in SIP Servlet Applications

To configure a highly available SIP Servlet application:

1. Modify the `sip.xml` file (located at `ORACLE_HOME/j2ee/ocms/applications/<application name>/<web module name>/WEB-INF/sip.xml`) to include the `<distributable>` element.

   For example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
     <sip-app>
     <display-name>proxyregistrarssr</display-name>
      <distributable/>
   <!--Servlets-->
   <servlet>
     <servlet-name>Registrar</servlet-name>
     <servlet-class>oracle.sdp.registrar.VoipRegistrarServlet</servlet-class>
     <init-param>
       <param-name>LocationService</param-name>
       <param-value>oracle.sdp.locationdbservice.LocationDbServiceBD
       </param-value>
      </init-param>
     </servlet>
   </sip-app>
   ```

2. Modify the `web.xml` file (located at `ORACLE_HOME/j2ee/ocms/applications/<application name>/<web module name>/WEB-INF/web.xml`) to include the `<distributable>` element.

   For example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
   2.2//EN" "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
   <web-app>
       <display-name>proxyregistrarssr</display-name>
       <distributable/>
   </web-app>
   ```

3. Modify the `orion-application.xml` file (located at `ORACLE_HOME/j2ee/ocms/application-deployments/<application name>/orion-application.xml`) to include the `<cluster>` element which is

used to configure clustering both for Oracle Application Server nodes as well as specific SIP servlet applications.

For example:

```
<orion-application ... >
   <cluster allow-colocation="false">
   ...
   </cluster>
</orion-application>
```

The `<cluster>` element, which is used in both `orion-application.xml` and `application.xml` files, includes the following sub-elements that control application replication:

– `enabled`—Specifies whether clustering is enabled. The default value is `true`.

– `group-name`—The name to use when establishing replication group channels. If not supplied, the application name as defined in `server.xml` (the Oracle Application Server configuration file) is used by default. New group channels are created for each enterprise application. If a value is specified, the application and all child applications use the channels associated with this group name.

– `allow-colocation`—Specifies whether to allow application state to be replicated to a node residing on the same host machine. The default value is `true`.

> **Note:** Although the default value is `true`, set `allow-colocation` to `false` if multiple hosts are available. If multiple Oracle Application Server instances are instantiated on the same machine, specify different listener ports for each instance in the `default-web-site.xml`, `jms.xml`, and `rmi.xml` configuration files.

– `write-quota`—The number of other group members to which the application state should replicate to. This attribute enables reducing overhead by limiting the number of nodes to which state is written. The default value is `1`.

For additional information regarding the `<cluster>` element and its sub-elements, refer to the chapter "Configuring Application Clustering" in *Containers for J2EE Configuration and Administration Guide*.

> **Note:** Ensure that you use the correct spelling for attributes in the `orion-application.xml` file, as misspellings will not result in error messages. For example, if you misspell `start-port` in the cluster configuration section as `start-prt`, replication will appear to have started even though session replication does not work.

4. Repeat these steps for each application deployed on each OCMS instance.

> **Important:** Deploy the application symmetrically to all SIP application server nodes.

For information about developing highly available SIP Servlet applications, refer to *Oracle Communication and Mobility Server Developer's Guide*.

Configuring Application Session Data Replication

OCMS supports multicast replication by defining the `orion-application.xml` file's `<cluster>` and `<property-config>` elements. The `<property-config>` element contains data required to use the JGroups group communication protocol to replicate session state across nodes in the cluster.

To set the replication policy, edit the `ORACLE_HOME/j2ee/ocms/application-deployments/<application name>/orion-application.xml` file as follows:

- Set the `<cluster>` element's `allow-colocation` attribute to `false`.

- Set the the `<property-config>` element's `<url>` element to the path to the JGroup's XML file describing the application-related high-availability configuration (illustrated in Example 8–4).

Example 8–3   Editing orion-application.xml File for Replication

```
<orion-application ... >
  ...
  <cluster allow-colocation="false">
  ...
    <property-config>
       <url>file:///ORACLE_HOME/j2ee/ocms/application-deployments/
       <application name>/jgroups-tcp.xml</url>
    </property-config>
  </cluster>
```

Example 8–4 illustrates the JGroups XML file (referred to as `jgroups-tcp.xml`) in Example 8–3.

Example 8–4   A Sample JGroups Application High Availability Configuration File

```
<config>
  <TCP/>
  <MPING mcast_addr="230.0.0.130" mcast_port="8500" ip_ttl="1"/>
  <MERGE2 min_interval="5000" max_interval="10000"/>
  <FD timeout="1000" max_tries="3" shun="false"/>
  <FD_SOCK/>
  <VERIFY_SUSPECT timeout="1000"/>
  <pbcast.NAKACK gc_lag="100" retransmit_timeout="3000"/>
  <pbcast.STABLE desired_avg_gossip="20000"/>
  <pbcast.GMS join_timeout="3000" join_retry_timeout="2000" shun="false"
  print_local_addr="true"/>
</config>
```

In Example 8–4, failure detection (set by `<FD timeout>`) is set in milliseconds. In the sample JGroups file, it is set at 1000 milliseconds with three retries (`max_tries="3"`).

Configuring High Availability for a Deployed SIP Servlet Application

Perform the following if the SIP Servlet application has already been developed and deployed, but not configured for high availability.

1. Undeploy the SIP Servlet application.

2. Unpack the application EAR.

3. Modify the `sip.xml` and `web.xml` files to include the `<distributable>` element, as described in "Enabling High Availability in SIP Servlet Applications". To edit the `sip.xml` and `web.xml` files, do the following:

   a. Create a new folder and name it `<Your SIP application>`.

   b. Unpack the EAR file to the new folder.

   c. Unpack the WAR file inside the EAR file to the folder `<Your SIP application>/<WAR module name>`.

   d. Edit the following files:

   ```
   <Your SIP application>/<WAR module name>/WEB-INF/sip.xml
   <Your SIP application>/<WAR module name>/WEB-INF/web.xml
   ```

   e. Under `<Your SIP application>/<WAR module name>`, re-package the contents of the WAR file using the original WAR file name. Replace the original WAR file with in the `<Your SIP application>` folder with the new one that you just created.

   f. Delete the `<Your SIP application>/<WAR module name>` folder.

   g. Package the contents of the `<Your SIP application>` into an EAR file. Use the file name of the original EAR file.

4. Re-deploy the application EAR to the SIP Servlet Container.

Disabling High Availability at the Application Level

To remove an application from the cluster:

1. In a text editor, open the application-specific `orion-application.xml` file.

2. Set the `enabled` attribute of the `<cluster>` element to `false`. For more information, see "Configuring Application Session Data Replication".

   For example:

   ```
   <orion-application ... >
      ...
      <cluster enabled="false">
      ...
      </cluster>
   </orion-application>
   ```

3. Save the `orion-application.xml` file.

Upgrading SIP Servlet Applications in OCMS

SIP Servlet applications can be upgraded using a rolling upgrade procedure that minimizes downtime.

To perform a rolling upgrade of a SIP servlet application in OCMS:

1. On the OCMS node where you want to upgrade the SIP Servlet application, execute the following command to stop running all OCMS processes on the node:

   ```
   ORACLE_HOME/opmn/bin/opmnctl shutdown
   ```

2. Comment out the `<topology>` element in the `ORACLE_HOME/opmn/conf/opmn.xml` file to remove the SIP Application Server from the cluster.

3. Restart the SIP container by running the following command:

   ```
   ORACLE_HOME/opmn/bin/opmnctl startall
   ```

4. Upgrade the SIP Servlet application on the SIP container you took out of the cluster.

5. Shut down the SIP container again so that you can put it back in the cluster:

   ```
   ORACLE_HOME/opmn/bin/opmnctl shutdown
   ```

6. Uncomment the `<topology>` element in the `opmn.xml` file to place the SIP container back into the cluster.

7. Place the SIP container back into the cluster by executing the following command:

   ```
   ORACLE_HOME/opmn/bin/opmnctl startall
   ```

   The SIP Servlet application upgrades following the completion of all in-progress calls.

8. Repeat the process with the remaining SIP containers.

Configuring the Proxy Registrar for High Availability

Configuring the Proxy Registrar for high availability involves the following main steps:

- Configuring the `orion-application.xml` file for the Proxy Registrar application, as described in "Configuring Application Session Data Replication"
- Configuring Oracle TimesTen Replication

Configuring Oracle TimesTen Replication

For each OCMS node in the cluster, Oracle TimesTen and OCMS must be installed and run under the same non-root user name at the operating system level. For example, if OCMS and TimesTen are installed by a user called *ocmsuser* on OCMS Node 1 of a cluster, then *ocmsuser* must install both OCSM and TimesTen on OCMS Node 2 as well.

Replicating an Oracle TimesTen In-Memory database in OCMS involves the following:

- Creating and Seeding Replication Tables—In the TimesTen database, create replication-related tables. Then seed the database with a list of tables that require replication.

- Creating an Internal TimesTen User—In the TimesTen database, create a user called *ocmsuser* as well as a password for that user. You then grant administrative privileges to *ocmsuser*.

- Configuring Replication in Oracle TimesTen In-Memory Database—Configure clusters and instances, including running a replication script followed by a database duplication script.

- Configuring Failover and Recovery for Oracle TimesTen In-Memory Database—Configure recovery procedures in the event of a failed instance.

   > **Note:** Oracle TimesTen database can be replicated synchronously or asynchronously. The current version of OCMS supports synchronous replication only.

Creating and Seeding Replication Tables

For **each OCMS node** in the cluster, perform the following:

1. Run the following Perl scripts:

```
$TT_HOME/bin/ttIsql -connStr "DSN=ocmsdb" -f
$ORACLE_HOME/sdp/replication/sql/replication.drop.timesten.sql
$TT_HOME/bin/ttIsql -connStr "DSN=ocmsdb" -f
$ORACLE_HOME/sdp/replication/sql/replication.timesten.sql
```

The following tables are created in the TimesTen database:

```
clusters
    cluster_id INT NOT NULL,
    cluster_name VARCHAR(255) NOT NULL UNIQUE
    cluster_replication VARCHAR(64) NOT NULL,
instances
    instance_id INT NOT NULL,
    instance_name VARCHAR(255) NOT NULL UNIQUE,
    hostname VARCHAR(255) INLINE NOT NULL,
    port INT NOT NULL
cluster_instance_map
    cluster_map_id INT NOT NULL,
    cluster_id INT NOT NULL,
    instance_id INT NOT NULL
replication_params
    replication_param_id INT NOT NULL,
    replication_param_name VARCHAR(255) NOT NULL UNIQUE,
    replication_param_value VARCHAR(255) NOT NULL,
replication_tables
    table_id INT NOT NULL,
    table_name VARCHAR(255) NOT NULL UNIQUE
replication_sequences
    account_seq
    role_seq
    credentials_seq
    realm_seq
    private_identity_seq
    property_seq
    public_identity_seq
    lsid
```

2. Modify `ORACLE_HOME/sdp/replication/sql/seed_clusters_test_sync.sql` to suit your deployment. By default, this script configures one cluster of two OCMS nodes. Replace *host1.example.com* and *host2.example.com* with the hostnames of the two OCMS nodes in the cluster.

3. Execute the following command:

```
TT_HOME/bin/ttIsql -connStr "DSN=ocmsdb" -f
ORACLE_HOME/sdp/replication/sql/seed_clusters_test_sync.sql
```

This command seeds the TimesTen database with all the cluster and instance information required for configuring replication.

Following are the tables that require replication:

- Schema properties:

  properties

- Cluster and instance configurations:

```
clusters
instances
cluster_instance_map
replication_params
replication_tables
```

- Location Service

```
locationservice
```

- User Service

```
public_identity
private_identity
```

- Security Service

```
account
role
user_role
credentials
realm
```

Note: Sequences are excluded from replication.

Creating an Internal TimesTen User

For **each OCMS node** in the cluster, perform the following:

1. Create an internal TimesTen user called *ocmsuser* and assign a password to the user name by executing the following command in TimesTen:

```
Command> create user ocmsuser IDENTIFIED BY 'password';
```

2. Grant *ocmsuser* administrative privileges by executing the following command in TimesTen:

```
Command> grant admin to ocmsuser;
```

Configuring Replication in Oracle TimesTen In-Memory Database

---

Note: It is only possible to configure replication between like operating systems, for example two Linux computers.

This section describes how to configure replication in the Oracle TimesTen in-memory database instances:

- Configuring Replication in the First Oracle TimesTen Database Instance in the Cluster
- Configuring Replication in the Second Oracle TimesTen Database Instance in the Cluster

Configuring Replication in the First Oracle TimesTen Database Instance in the Cluster To configure replication for the first Oracle TimesTen In-Memory database in a cluster, perform the following:

1. On each computer, add *$TT_HOME/lib* to the *LD_LIBRARY_PATH* environment variable.

2. On the first computer in the cluster, execute the following command:

```
$ORACLE_HOME/sdp/replication/bin/create_replication_sync.pl
```

3. When prompted for the hostname, enter the fully qualified domain name.

The following error message may display. You can safely ignore it.

```
[TimesTen][TimesTen 6.0.7 ODBC Driver][TimesTen]TT12027: The agent is already
stopped for the data store.
Dropping Replication Scheme: ocms_rep.repscheme
[TimesTen][TimesTen 6.0.7 ODBC Driver][TimesTen]TT8160: REPLICATION OCMS_
REP.REPSCHEME not found
        -- file "repDDL.c", lineno 1436, procedure "sbRepCheckOrDrop()"
```

The replication scheme is now configured on the database.

4. Verify that the replication scheme has been configured in the data store by executing the following command at the TimesTen prompt:

```
$TT_HOME/bin/ttisql -connstr "dsn=ocmsdb"
Command> repschemes;
```

5. Verify that replication scheme OCMS_REP.REPSCHEME has been successfully created.

6. If you need to make any changes to the names of the TimesTen instances, for example, if you opt to replicate to a different instance altogether, perform the following:

    a. Drop and re-create the replication tables by executing the following commands:

```
$TT_HOME/bin/ttisql -connstr "dsn=ocmsdb"
Command> drop replication ocms_rep.repscheme;
Command> quit
```

    b. Go back to "Creating and Seeding Replication Tables".

7. Configure TimesTen to automatically start replication (as well as the replication agent, if it is not running) upon restarting TimesTen by executing the following command:

```
$TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
```

Configuring Replication in the Second Oracle TimesTen Database Instance in the Cluster To configure replication for the second Oracle TimesTen Database in the cluster:

1. If OCMS is running, stop running OCMS by executing the following command:

```
opmnctl stopproc process-type=ocms
```

2. Stop running Oracle TimesTen by executing the following command:

```
opmnctl stopproc ias-component=TIMESTEN
```

3. Start the TimesTen daemon by executing the following command:

```
$TT_HOME/bin/ttdaemonadmin -start
```

4. Configure the ocmsdb data store to have the same PermSize as that of the first computer. For example, if the first computer has 512 MB PermSize allocated to ocmsdb, allocate 512 MB to the second ocmsdb data store.

On Linux, edit `$TT_HOME/info/sys.odbc.ini` and change the PermSize corresponding to the ocmsdb entry.

5. Delete the ocmsdb datastore on this computer by executing the following command:

```
ttdestroy -connstr "dsn=ocmsdb"
```

The datastore will be replicated from the first OCMS computer.

6. Duplicate the ocmsdb datastore from the first computer as follows:

   a. Execute the following command:

   ```
   perl $ORACLE_HOME/sdp/replication/bin/duplicate_db_replication.pl
   ```

   b. At the prompt, enter the fully qualified domain name of the first computer.

   c. At the prompt, enter the fully qualified domain name of the second computer (that is, the computer on which you are currently working).

   d. At the prompt, enter the user name and password of the TimesTen user that you created on the first computer, for example, *ocmsuser*.

   > **Note:** Depending on its size, copying and replication of the datastore may take a few moments.

7. Start the TimesTen daemon as follows:

```
$TT_HOME/bin/ttdaemonadmin -start
```

8. Run the following command to configure replication to start automatically when TimesTen restarts:

```
$TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
```

9. Start TimesTen using opmnctl:

```
opmnctl startproc ias-component=TIMESTEN
```

10. Start OCMS:

```
opmnctl startproc process-type=ocms
```

Configuring Failover and Recovery for Oracle TimesTen In-Memory Database

This section describes the following topics:

- Configuring Failover
- Configuring Recovery

Configuring Failover TimesTen is managed by OPMN, such that OPMN immediately revives a failed instance. The replication policy is configured to auto-start when TimesTen starts up and no further action is required. If the replication agent fails, TimesTen immediately restarts it.

As any given application instance is tied to an in-memory instance of the TimesTen database, the application instance cannot fail over to another instance of the TimesTen database.

Typically, when an application instance fails, requests are no longer sent to the failed instance. No action is required on the other instances in the cluster. When the application instance comes back online, however, a recovery procedure must be followed.

Configuring Recovery If you have manually brought down an instance for a few minutes, you can simply restart it. The replication policy is configured to auto-start when TimesTen starts, such that no further action is required.

If an instance has failed for an extended period of time (over an hour, for example), it is recommended to delete the database on the failed instance and duplicate it from any another instance in the cluster.

To recover a failed instance, perform the following:

1. Reset the replication policy to manual, so that the replication agent can terminate properly.

   ```
   $TT_HOME/bin/ttAdmin -repPolicy manual ocmsdb
   ```

2. Stop replication on the failed instance.

   ```
   $TT_HOME/bin/ttAdmin -connstr "dsn=ocmsdb" -repStop
   ```

3. Run the following Perl script to recover replication:

   ```
   perl $ORACLE_HOME/sdp/replication/bin/recover_replication_sync.pl
   ```

4. Change replication policy back to *always* by executing the following command:

   ```
   $TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
   ```

On the other OCMS node (that is, the surviving instance), perform the following:

1. Using the following command, change the replication policy to *manual* to enable the replication agent to terminate proplery:

   ```
   $TT_HOME/bin/ttAdmin -repPolicy manual ocmsdb
   ```

2. Stop replication on the surviving instance:

   ```
   $TT_HOME/bin/ttAdmin -repStop ocmsdb
   ```

3. Drop the replication scheme. (This is executed using the `drop replication` command).

   ```
   $TT_HOME/bin/ttisql -connstr "dsn=ocmsdb"
   Command> drop replication ocms_rep.repscheme;
   Command> quit
   ```

4. Recreate replication:

   a. Execute the following command:

      ```
      perl $ORACLE_HOME/sdp/replication/bin/create_replication_sync.pl
      ```

      > **Note:** You must add *TT_HOME/lib* to the *LD_LIBRARY_PATH* environment variable before you can execute the `create_replication_sync.pl` command.

    b. When prompted for the hostname, enter the fully qualified domain name of the host.

5. Set the replication back to *always* by executing the following command:

```
$TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
```

Troubleshooting Replication

This section discusses the following replication troubleshooting issues:

- Unable to Connect
- Master Catchup Required

Unable to Connect When attempting to duplicate the database from the first instance to the second instance, the following error message might display:

```
Unable to connect to the replication agent on the first instance
```

This message indicates that the replication agent may not be running.

To resolve this error, perform the following:

- Ensure that the replication agent is running on the first database instance by executing the following command:

  ```
  $TT_HOME/bin/ttStatus
  ```

  The output of this command should include entries for replication with corresponding process IDs.

- The replication policy might not be configured as always on. To configure the replication policy as `always`, execute the following command:

  ```
  $TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
  ```

Master Catchup Required When attempting to connect to the database on the first instance, the following error message may display:

```
Master Catchup Required. New Connections are not allowed.
```

To resolve this error, drop and re-create the replication scheme as follows:

1. On the first instance, edit `$TT_HOME/info/sys.odbc.ini` and set `ForceConnect=1` for the ocmsdb data store. This enables connections and modifications to the database.

2. Next, you must drop the existing replication scheme, create a new replication scheme, and re-create the replication sequences. To do this, execute the following commands:

```
$TT_HOME/bin/ttAdmin -repPolicy manual ocmsdb
$TT_HOME/bin/ttAdmin -repStop -connstr "dsn=ocmsdb"
perl $ORACLE_HOME/sdp/replication/bin/create_replication_sync.pl
```

3. Execute the following command to configure the replication policy as `always` on:

```
$TT_HOME/bin/ttAdmin -repPolicy always ocmsdb
```

4. Repeat for the other database instance, then duplicate the database from the first instance to the second.

9

Managing the SIP Server

This chapter, through the following sections, describes how to manage the OCMS SIP Server through Oracle 10*g* Enterprise Application Server Control.

- "Overview of SIP Server Management"
- "Starting, Stopping and Restarting the OCMS SIP Server"
- "Managing OCMS MBeans"
- "Deploying SIP Server Applications"
- "Viewing Log Files"
- "Managing OCMS through JBoss Application Server"

Overview of SIP Server Management

The Oracle SIP Server is an OC4J container that you manage using the Oracle 10*g* Enterprise Manager Application Server Control console (Figure 9–1).

Figure 9–1   The Oracle 10g Enterprise Manager Application Server Control
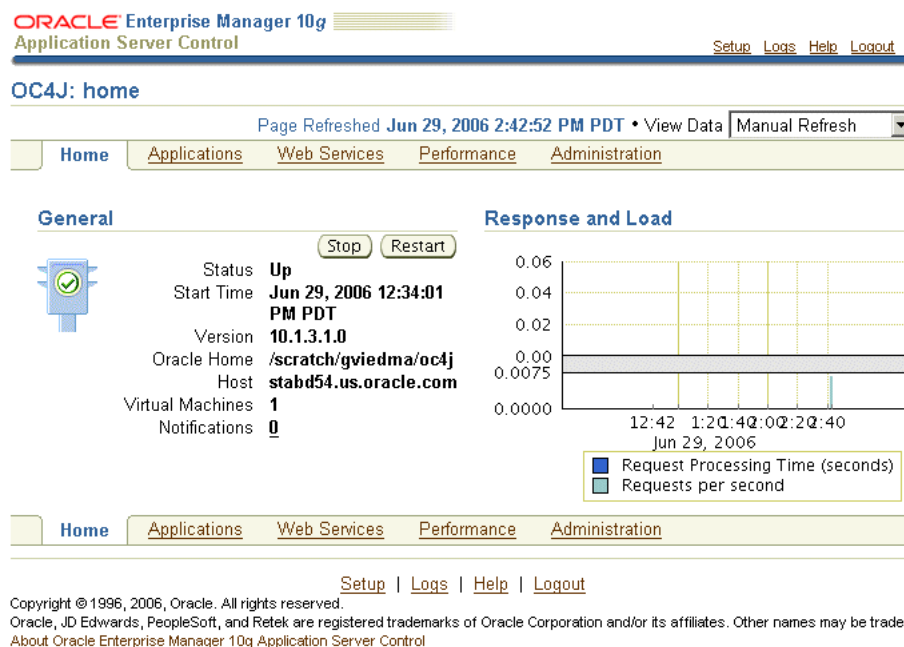
In addition to the standard Application Server Control functions that enable starting, stopping, restarting, deploying, undeploying, and redeploying applications, the Application Server Control MBean browser enables you to configure and manage the OCMS components. Configuring the attributes of the OCMS MBeans (Managed Beans) enables you to execute administrative tasks and set the basic configuration (port, IP, and host address) of the OCMS SIP Server itself. In addition, the OCMS MBeans enables you to configure and manage presence.

Starting, Stopping and Restarting the OCMS SIP Server

Application Sever Control enables you to stop and restart The Oracle SIP Server. Like other OC4J containers, the OCMS SIP Servlet can be stopped and restarted using the **Stop** and **Restart** buttons on the *Home* page (Figure 9–1). These buttons control the entire OC4J instance; stopping or restarting OC4J also stops or restarts the OCMS SIP Servlet Container and the applications deployed to it. See also "Starting the SIP Servlet Container".

From the command line, you can invoke the start and stop scripts available under ORACLE_HOME/sdp/bin.

You can also stop or restart OC4J using the `opmnctl` or the `admin_client.jar` command-line utility to stop, start, or restart OC4J or its applications.

To stop all OPMN-managed processes including OC4J on a local Oracle Application Server instance using `opmnctl`:

```
opmnctl stopall
```

To stop the OC4J instance of OCMS on a local Oracle Application Server instance using `opmnctl`:

```
opmnctl stopproc process-type=ocms
```

To start all OPMN-managed processes, including OC4J, on a local Oracle Application Server instance using `opmnctl`:

```
cd ORACLE_HOME/opmn/bin
opmnctl startall
```

To restart the OC4J instance of OCMS on a local Oracle Application Server instance using `opmnctl`:

```
cd ORACLE_HOME/opmn/bin
opmnctl restartproc process-type=ocms
```

Use the following syntax when using `admin_client.jar` to start, stop or restart an application and its child applications on a specific OC4J instance or across an entire cluster.

```
java -jar admin_client.jar uri adminId adminPassword -start|-stop appName
```

For more information, see *Oracle Application Server Administrator's Guide*.

Starting an Application and Stopping a SIP Servlet Application

The **Stop**, **Start** and **Restart** buttons on the *Applications* page (Figure 9–2) control the running status for selected SIP servlet applications.

Figure 9–2   The Applications Page of the Application Server Control

> **Note:**   Changes made to the SIP Container applications persist when you restart OC4J; changes made to the logging do not persist.

Managing OCMS MBeans

The Application Server Control Console's MBeans browser (Figure 9–3) enables you to view and configure MBeans. An MBean (managed bean) is a Java object that represents a JMX-manageable resource in a distributed environment, such as an application, a service, a component, or a device. MBeans expose a management interface, including a set of attributes and operations. This interface does not change during the lifetime of an MBean instance. MBeans can also send notifications when defined events occur.

The MBean attributes enable you to configure the OCMS SIP Server.

Figure 9–3   Viewing MBeans

See *Oracle Containers for J2EE Configuration and Administration Guide* for information on managing MBeans.

Accessing MBeans

The Application Server Control Console's MBean browser enable you to view, configure and deploy both system MBeans and application-defined MBeans.

To display the available System MBeans, the MBean Browser accesses the MBean Server that runs in OC4J. The available System MBeans display in the System MBean Browser.

In general, you access the MBeans related to the JSR 116-compliant OCMS SIP Servlet Container, a runtime environment for SIP applications that includes security, concurrency, life-cycle management, transaction, and other services through the System MBean Browser. For more information, see "Managing OCMS MBeans".

Figure 9–4   Viewing System MBeans
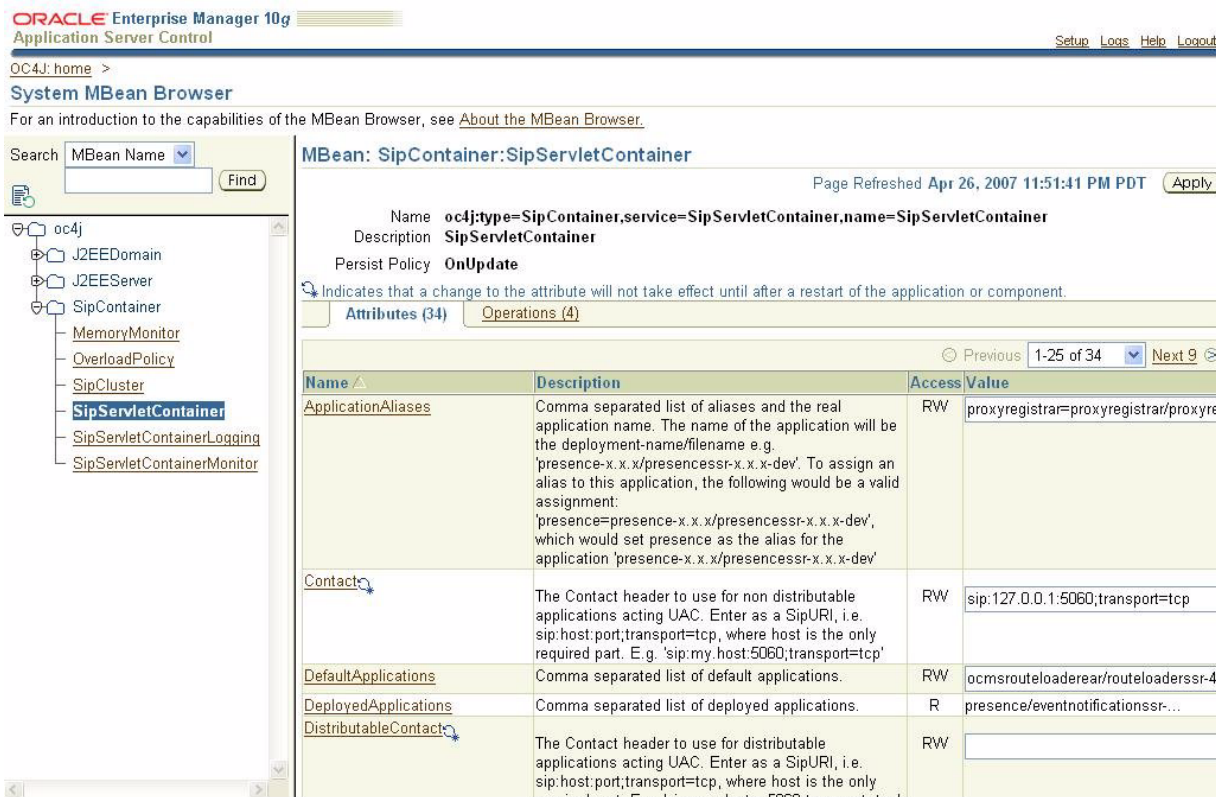
The SIP Application-based MBeans do not display in the System MBean Browser; instead, they appear in the context of the application that registered them. For example, Figure 9–3 illustrates the MBeans registered by the Subscriber Data Services application.

> **Note:**   MBeans are packaged with the application that they manage.

Accessing SIP Servlet Container MBeans

The System MBean Browser enables you to browse the SIP Servlet Container MBeans (Figure 9–3). To access the System MBean Browser:

1. Navigate to the OC4J Home page for the SIP Server instance.

2. Click **Administration**. The *Administration* page appears, displaying the available tasks.

3. If needed, expand the JMX section of the task list to display *System MBean Browser*.

4. Click the *Go to Task* icon in the *System Bean Browser* row of the table. The System MBean Browser appears, displaying the available MBeans in the tree control. Selecting an MBean in the tree control enables you to view and edit its attributes, invoke its operations, and manage notification subscriptions. Click **Apply** to commit any changes to the MBean's parameters.

Figure 9–5   The JMX Section of the Task List

> **Tip:**   Use the *Search* function to locate an MBean

Accessing the MBeans for a Selected SIP Application

To view the MBeans for a selected application:

1. Click **Applications** on the SIP Server OC4J home page. A list of applications appears.

2. Click the *Application Defined MBean* icon for the selected application. The *Application MBeans* page appears. The tree control lists the MBeans registered to the application. Selecting an MBean in this tree control enables you to view its attributes, operations, statistics, and notifications.

> **Note:**   The MBean attributes represent the getter and setter methods of the MBean operations.

3. Click **Apply** to commit any changes to the MBean's attributes.

Configuring the SIP Servlet Container MBeans

This section describes the configuration parameters for the following System MBeans (listed in Table 9–1):

Table 9–1   SIP Servlet Container MBeans

Tasks	MBean Name
Tasks include: ■ Setting or changing the IP address of the SIP Container. ■ Setting or changing the DNS IP address. ■ Setting or changing the domains and realms. ■ Setting or changing the traffic port. ■ Setting or changing the proxy that receives client requests. ■ Setting the default applications for incoming requests.	SIP Servlet Container
Setting the log levels for the logger components (*badmsg*, *config*, *eventlogger*, *statistics*, *system*, and *traffic*). For more information, see Chapter 10, "Configuring the Logging System".	SIP Servlet Container Logging
Setting bindings that enable clients to traverse NAT (Network Address Translating) entities.	STUN Service
View the current statuses of the system queues.	SIP Servlet Container Monitor
Set overload actions when the capacity thresholds for memory usage, Application Queue usage, Network Queue usage, or SIP Session Table usage are reached.	Overload Policy
Set the default timeouts for OC4J clusters.	SIP Cluster

SIP Servlet Container

The SIP servlet container is a standalone Java process that provides the execution environment for the SIP applications. The attributes exposed by the SIP Servlet Container MBean (SipServletContainer) enable you to change values that are set during the installation of OCMS (listed in Table 9–2). For more information on setting the values for these attributes during installation, refer to *Oracle Communication and Mobility Server Installation Guide*.

Table 9–2   Values Populated by Installation of OCMS

Value	SIP Servlet Container Attribute(s)
The IP address of the SIP Container.	IPAddress
The traffic port used by the SIP Container.	SIPPort
The proxy for receiving client requests.	EdgeProxy
The domain (or hostname) of the SIP Server and the SIP realm used for authentication.	DomainsandRealms

Table 9–3 describes the SIP Servlet Container MBean's attributes. While the format for entering values is *sip:host:port;transport=tcp|udp* for the following attributes, you generally need only enter the host value, as other values are pre-seeded.

■ Contact

- DistributableContact

- DistributableRecordRoute

- DistributableVia

- DnsIpAddress

- DnsIpAddress

- Via

Table 9–3   Attributes of the SIP Servlet Container

Attribute	Value
ApplicationAliases	A short name for the application. For more information, see "Setting an Alias for an Application".
Contact	The host, port, and transport used in the *Contact* header that is embedded in SIP requests by applications acting as User Agent Clients (UACs). Enter this value as a SIP URI. This contact information provides an address that enables the SIP Server to respond.
DefaultApplications	A comma-separated list of applications that are invoked if no application matches a request. By default, the Application Router is set as the default application. If many applications have been deployed to the SIP Container, you should also deploy an application dispatcher as the default application. The application dispatcher (or Application Router in OCMS) routes incoming requests to other applications by addressing them directly. For more information, see "Application Router" in "An Overview of Oracle Communication and Mobility Server". If no default application is set, then the container designates all of the SIP applications that it locates as the default application. For more information on default application and how the SIP container invokes servlets, see *Servlet Mapping* in *Oracle Communication and Mobility Server Developer's Guide*.
DeployedApplications	A read-only list of deployed applications.
DistributableContact	The host, port, and transport placed in the *Contact* header for distributable applications acting as User Agent Clients (UACs) in a high-availability environment.

DistributableRecordRoute	The host, port, and transport placed in the *RecordRoute* header for distributable applications in a high-availability environment. Enter this value as a SIP URI in the following format:
	
	`sip:host:port;transport=tcp.`
	
	You need only enter the hostname (`sip:my.host:5060;transport=tcp`).
	
	For high availability configurations, configure this parameter in the following format:
	
	`sip:<SIP container IP address>:<port>.`
	
	Remove the transport method to enable the use of any type of transport between the Edge Proxy and OCMS. For more information, see "Configuring the OCMS SIP Containers for High Availability".

DistributableVia	The host, port, and transport used in the *Via* header for distributable applications in a high-availability environment. Enter this value as a SIP URI in the following format: `sip:host:port;transport=tcp.` You need only enter the hostname (`sip:my.host:5060;transport=tcp`).

Table 9–3   (Cont.) Attributes of the SIP Servlet Container

Attribute	Value
DnsIpAddress	The IP address for the DNS against which the SIP servlet containers resolve addresses. Enter the IP address in the format of `<IP>:<port>/<transport>`. For example, enter *127.0.0.1:53/UDP*. You can only enter an IP address as the value; you cannot enter a domain name.
DomainsAndRealms	This attribute defines a mapping of SIP domains to Java-authenticated realms. This mapping is used to dynamically challenge a SIP request with the appropriate authentication realm during SIP authentication. For example, a domain/realm mapping of *voip.com/voip* would require a user with the SIP URI of *user@voip.com* to authenticate against the *voip* realm in response to the challenge by the SIP servlet container. Enter a comma-separated list of the configured hosted domains and their corresponding realms used for authentication. Applications -- rather than the SIP container itself -- use such a hosted domain to send a 404 (*Not Found*) message.
DomainLoopDetection	If set to *true* (the default setting), The SIP container checks whenever a request is about to be sent from the server to a destination that has a resolvable host name. If the host name resolves to one of the server's own listening ports (meaning that the server will send the request back to itself) the server checks the resolved host name against configurations set for the *DomainsandRealms* and *RecordRoute* header attributes. If there is no match for the hostname, then the request is blocked and the server responds with a 482 (*Loop Detected*) message. This protects the server from fake DNS names that point to the server's IP address and then send a request with the fake hostname in the top-most route header. Because the SIP container cannot recognize the hostname, it will never pop the Route header and the request will loop.
IPAddress	The IP address on which the SIP servlet container listens. The default value of 0.0.0.0 designates all IP addresses. For a production environment, change this value to an actual IP address.
NetworkThreadCount	The number of network threads spawned by the SIP container to handle network traffic. The value must be an integer.
Edge Proxy	The proxy that receives client requests. This proxy adds a pre-loaded route. For example: *sip:my.host:5060;transport=tcp*. This setting is required for clients to maintain a reusable TCP connection to the server. To ensure this connection, clients may implement a keep-alive algorithm, such as sending periodic CRLFs (carriage-return line feeds). In clustered environments, you must configure an Edge Proxy for each OCMS instance to support communication with the Edge Proxy or other proxy applications. For high availability installations, this attribute must be set to the address of the Edge Proxy or to the virtual hostname and port service used by the load balancer in front of the Edge Proxies. Set this value using the following format: `sip:<EdgeProxy or Load Balancer IP address>:<port>;lr` For more information, see "Configuring the OCMS SIP Containers for High Availability".

Table 9–3   (Cont.) Attributes of the SIP Servlet Container

Attribute	Value
RecordRoute	The host, port and transport used in the *RecordRoute* header of non-distributable applications. Enter this value as a SIP URI in the following format:
	`sip:host:port;transport=tcp.`
	You need only enter the hostname (`sip:my.host:5060;transport=tcp`).
	For high availability configurations, configure this parameter in the following format:
	`sip:<SIP Container IP address>:<port>.`
	Remove transports methods to enable any type of transport to be used between the Edge Proxy and OCMS. For more information, see "Configuring the OCMS SIP Containers for High Availability".
SIPPort	The port through which the SIP Servlet Container accepts traffic. The default value is 5060.
SipServletCommanInterceptors	A comma-separated list of SipServlet interceptor classes. The interceptors must implement `org.aopalliance.intercept.MethodInterceptor`. If the class has a `public` constructor with `javax.servlet.ServletConfig` it will be used to instantiate the interceptor. To enable P-Asserted-Identity support (RFC 3325) for authentication when using OCMS UserService and SecurityService, replace `oracle.sdp.sipservletengine.SecurityInterceptor` with `oracle.sdp.extinterceptors.PaiSecurityInterceptor`.
SipServletOc4jInterceptors	A comma-separated list of SipServlet interceptor classes specific to OC4J. The interceptors must implement `org.aopalliance.intercept.MethodInterceptor`. If the class has a `public` constructor with `javax.servlet.ServletConfig` it will be used to instantiate the interceptor. This list will be prepended to the list of interceptors entered for the *SipServletCommonInterceptors* attribute.
StatisticsPeriodicity	The interval, in minutes, at which the SIP servlet container logs statistics that display in the SIP Servlet Container Monitor. Setting this attribute to 0 suspends logging operation. See also the statistics attribute of the SIP Servlet Container Logging MBean.
TimerListenerOc4jInterceptors	A comma-separated list of OC4J-specific `TimerListener` interceptor classes. The interceptors must implement `org.aopalliance.intercept.MethodInterceptor`.
TimerT1	The estimate, in seconds, for a round trip. This value must be an integer.
TimerT2	The maximum interval, in seconds, for non-INVITE requests and INVITE responses. This value must be an integer.
TimerT4	The maximum interval, in seconds, that a message can remain in the network. This value must be an integer.
Trusted Hosts	A comma-separated list IP of addresses representing trusted hosts as described in RFC 3325. For example, enter 192.168.0.10, 192.168.0.11. Leaving the value field blank means that there are no trusted hosts; entering an asterisk (*), means that all IP addresses can be trusted. Regular expressions are not supported.
UdptoTcpTriggerSize	The maximum number of bytes contained in a request message sent by the SIP servlet container. When a request message reaches this size, the SIP servlet container uses TCP rather than UDP to transport the request. The default value for a request is 1300 bytes.
UseStun	Select *true* to use STUNbis keepalive traffic.

Table 9–3   (Cont.) Attributes of the SIP Servlet Container

Attribute	Value
UseTCP	Select *true* for the SIP servlet container to listen for TCP traffic. Oracle recommends setting TCP as the transport protocol because of network fragmentation issues. Add NAPTR and SRV records to the DNS to indicate that TCP is the preferred protocol. Ensure that clients connecting to OCMS fully support NAPTR and SRV records.
UseUDP	Select *true* for the SIP servlet container to listen for UDP traffic.
Via	The host, port, and transport used in the *VIA* header of non-distributable applications. Enter this value as a SIP URI.

Setting an Alias for an Application

A SIP application in the OCMS SIP servlet container is basically the parsed `sip.xml` file. Hence, the SIP application contains all of the parsed servlet-definitions as well as the servlet-mappings (that is, the rules) and other `sip.xml` components as listeners.

As illustrated in Chapter 7, "Packaging and Deploying Applications", The SIP application is usually built as an Enterprise Archive (EAR) that is deployed to OC4J. Once the SIP application has been deployed, the application server parses the `sip.xml` and instantiates the servlets and listeners. The application server also creates a name for the deployed application by piecing together the components of the EAR file. Because this generated name can be lengthy, you can create an alias for the application. You can use this shorter, more intuitive application name when you configure the default application. This short name can be used in the URI parameter of the route header or the request-URI.

For example, the name of a presence application deployed to the server might appear as *presenceapplicationear-4.0.0-dev/eventnotificationssr-4.0.0-dev*. Using the *ApplicationAliases* attribute, you can set a short name for this application by entering key-value pair such as *presence=presenceapplicationear-4.0.0-dev/eventnotificationssr-4.0.0-dev*. The key is the alias (*presence*) and the value is the real name of the application. This alias can be used for the real name of the application.

> **Note:**   The value for set for *ApplicationAliases* attribute must also match the `appId` parameter of the the Application Router's *SIPURIList* attribute to ensure that the OCMS SIP servlet container invokes the applications that are appropriate to incoming requests. The `appId` parameter is case-sensitive.

SIP Servlet Container Logging

The SIP Servlet Container Logging (SipServletContainerLogging) MBean's attributes (described in Table 9–4) expose the log level interfaces of Apache log4j (that is, the `level` class that extends `org.apache.log4j.priority`). These attributes enable you to set the following logging levels for SIP traffic and events, and also SIP servlet container's internal system logs:

- *OFF* – The *OFF* level turns off all logging. This is the highest log level.
- *ERROR* – The *ERROR* level designates error events which may allow the application to continue running. *ERROR* messages indicate a serious problem that requires immediate attention. *ERROR* messages are not caused by bugs in OCMS.
- *WARN* – The *WARN* level designates potentially harmful situations.

- *INFO* – The *INFO* level designates informational messages that highlight the progress of the application at a general level. These messages indicate a major life-cycle event, such as the activation or de-activation of a primary sub-component. This is the default log level set for all loggers except for *badmsg* and *traffic*.

- *DEBUG* – The *DEBUG* Level designates detailed informational events which are useful in debugging an application. These messages indicate trace or debug information for events that are meaningful to end users of the product, such as public API entry or exit points.

- *ALL* – The *ALL* level turns on all logging. This is the lowest log level.

> **Note:** Some of the component loggers do not use all of these logging levels.

By default, all of the component loggers except for *traffic* and *badmsg* are set to *INFO*. The *traffic* and *badmsg* logs are set to *OFF*. As a consequence, a system using these default settings will not write any messages to the *traffic* or *badsmg* logs.

Table 9–4   Attributes of the SipServletContainerLogging MBean

Attribute	Description
badmsg	Sets the log level for the bad messages log. This is a debug log.
config	Sets the log level for the configuration log. This is a debug log.
eventlogger	Sets the log level for the event log. You can use the event logs for charging.
statistics	Sets the log level for the *statistics* log, a debug log that reports SIP Servlet Container Monitor- related statistics at regular intervals. See also the SIP Servlet Container's StatisticsPeriodicityattribute.
system	Sets the log level for all SDP-related events.
traffic	Sets the log level for SIP traffic. This is a debug log.

For information on configuring the log4j logging system, see "Configuring the Logging System". For information on viewing the log files, see "Viewing Log Files".

STUN Service

The OCMS 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. OCMS 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 9–5) 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 9–5   Attributes of the STUNService MBean

Attribute	Value
Autostart	Set to *true* for the Stun Server to start automatically when OCMS 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.

SIP Servlet Container Monitor

The SIP Servlet Container Monitor MBean (SipServletContainerMonitor) displays read-only values for system queues and SIP transactions that enable you to assess the performance of the SIP servlet container when it enters an abnormal state. These values can also serve as a reference for system tuning.

The following sections describe SIP Servlet Container Monitor's attributes:

- "Viewing System Status"

- "Viewing Transactions"

- "Using the Current, Peak, and Total Usage Statistics to Tune the System"

Viewing System Status

Table 9–6 lists the attributes that display the current status of the SIP servlet container.

Table 9–6

SIP Servlet Container Status	Attribute
The time that the SIP container was last started.	SipServiceStartTime
The time that the SIP container entered its current state.	SipServiceLastChanged
The number of events dropped by OCMS.	SipMessagesDropped

Viewing Transactions

The SipServletContainerMonitor MBean's *SipSummaryTotalTransactions* attribute displays the total number of transactions (both current and completed) as a read-only value. The MBean further delineates transactions by displaying the total number of requests received by the SIP servlet container as well as the total number of subsequent responses that it receives and sends.

Total Requests and Responses

The *SipSummaryInRequests* attribute displays the total number of requests received by the SIP servlet container. In addition to this attribute, the SipServletContainerMonitor provides read-only values for the responses that comprise each transaction, both in terms of the total of SIP response messages received and sent by the SIP servlet container (the *SipSummaryInResponses* and *SipSummaryOutResponses* attributes, respectively) and also in terms of the total number of messages from provisional responses (Status Code 1xx) to the final responses (Status Codes 200 - 600). The SipServletContainerMonitor displays the total of SIP responses for each message category supported by OCMS (Table 9–7). See RFC 3261 and RFC 3265 for more information on response codes.

Table 9–7   SIP Response Messages Supported by OCMS

Status Code	Reason Phrase	Related Attributes
1xx	Provisional Response Messages	1xx-related attributes: ■ SipStatsInfoClassIns ■ SipStatsInfoClassOuts
2xx	Success Messages	2xx-related attributes: ■ SipStatsSuccessClassOuts ■ SipStatsSuccessClassIns
3xx	Redirection Response Messages	3xx-related attributes: ■ SipStatesRedirClassIns ■ SipStatsRedirClassOuts
4xx	Client Error Responses	4xx-related attributes: ■ SipStatsReqFailClassIns ■ SipStatsReqFailClassOuts
5xx	Server Failure Responses	5xx-related attributes: ■ SipStatsServerFailClassIns ■ SipStatsServerFailClassOuts ■ 503ResponseSent
6xx	Global Failure Messages	6xx-related attributes: ■ SipStatsGlobalFailClassIns ■ SipStatsGlobalFailClassOuts
NA	Non-SIP Response Codes	Non-SIP response-related attributes: ■ SipStatsOtherClassesIns ■ SipStatsOtherClassesOuts

Provisional Response Messages

As described in RFC 3261, the 1xx SIP response messages indicate informational or provisional responses and are sent when servers expects that obtaining a final response will exceed 200 milliseconds. The *SipStatsInfoClassIns* indicates the total number of 1xx responses received by the SIP servlet container, including transmission. The *SipStatsInfoClassOuts* attribute represents the number of messages sent, relayed, or re-transmitted by the SIP servlet container.

Success Messages

The *SipStatsSuccessClassOuts* and *SipStatsSuccessClassIns* represent the total number of 200 (OK) or 202 (Accepted) response messages sent and received by the SIP servlet container, respectively.

Redirection Response Messages

The 3xx responses provide information about the user's new location, or about alternative services that might be able to satisfy the call. The *SipStatesRedirClassIns* attribute represents the total number of 3xx responses received by (and re-transmitted to) the SIP servlet container. The *SipStatsRedirClassOuts* attribute represents the total number of 3xx responses sent (or re-transmitted) by the SIP servlet container.

Client Error Responses

The 4xx response messages are failure responses issued from server. When a client receives a 4xx response message, it should not attempt to send the request again without modifying it. The *SipStatsReqFailClassIns* attribute represents the total number of client error received by (or re-transmitted to) the SIP Servlet Container. The *SipStatsReqFailClassOuts* attribute represents the total number of client error messages sent (or retransmitted) by the SIP servlet container.

Server Failure Responses

The SIP Servlet Container sends the following error messages:

- 500 Server Internal Error
- 501 Not Implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Server Time-out
- 505 Version Not Supported
- 513 Message Too Large

The *SipStatsServerFailClassIns* attribute represents the total number of 5xx response messages received by (or re-transmitted to) the SIP servlet container. The *SipStatsServerFailClassOuts* attribute represents the total number of 5xx response messages sent (or re-transmitted) by the SIP servlet container.

While the *SipStatsServerFailClassOuts* attribute represents all of the 5xx response messages sent by the SIP servlet container, the *503ResponseSent* attribute represents the total number of 503 (*Service Unavailable*) responses sent by the SIP servlet container.

Global Failure Messages

The 6xx responses provide information specific to a user (as opposed to information specific to the instance indicated in the Request-URI). The *SipStatsGlobalFailClassIns* represents the total number of 6xx response messages received by (or re-transmitted)

to the SIP servlet container. The *SipStatsGlobalFailClassOuts* attribute represents the total number of 6xx responses sent (or relayed) by the SIP servlet container.

Non-SIP Response Codes

The *SipStatsOtherClassesIns* and *SipStatsOtherClassesOuts* represent non-SIP response messages (that is, response codes other than the 1xx, 2xx, 3xx, 4xx, 5xx, and 6xx that OCMS supports). The *SipStatsOtherClassesIns* attribute represents the total number of non-SIP responses messages received by (or re-transmitted to) the SIP servlet container. The *SipStatsClassesOuts* attribute represents the total number of non-SIP response messages sent (or relayed) by the SIP servlet container.

Using the Current, Peak, and Total Usage Statistics to Tune the System

This SipServletContainerMontior's attributes enable you to view the current, peak and total usage for the application and network queues as well as the current and total number of SIP sessions. These read-only values can serve as a reference when you tune overload protection using the Overload Policy MBean. In particular, you can use the numbers for *ApplicationPeakQueue*, *NetworkPeakQueue*, and *Sessions* to gauge the values for the Overload Policy's *AppQueueMaxSize*, *StackQueueMaxSize*, and *SipSessionTableMaxSize* attributes.

In general, overload protection is not invoked for normal load situations. Use this Mbean to find out the numbers for a normal load by monitoring the system's responses to various test scenarios without executing the overload protection. You can then set the overload protection to start above these figures. See also "Deactivating the Overload Protection for System Tuning".

> **Note:** The number of 503 responses sent and messages dropped (indicated by the *503ResponseSent* and *MessagesDropped* attributes, respectively) indicate how often overload protection should execute to reduce incoming traffic.

Overload Policy

The Overload Policy enables OCMS to execute overload protection when capacity reaches high threshold levels. To configure this MBean, you first set the maximum value for number of SIP sessions (*SipSessionTableMaxSize*) and the maximum size for the Application (*AppQueueMaxSize*) and Network queues (*StackQueueMaxSize*) as described in Table 9–8). You then set the threshold values for each of these, as described in Configuring High and Low Thresholds.

In addition to the maximum values settings, the *AllowedTrafficDuring503* attribute enables you to set the percentage of traffic allowed to pass through the system when it becomes overloaded and issues the 503 (*Service Unavailable*) error response to clients. The system does not process any new incoming requests if you set this attribute to zero (0), the default value.

> **Note:** Refer to the load figures displayed for the *ApplicationPeakQueue*, *NetworkPeakQueue*, *Sessions*, *503ResponseSent* and *MessagesDropped* attributes of the SIP Servlet Container Monitor when setting these values.

Table 9–8   Default Values for SIP Sessions, Application and Network Queues

Attribute	Default Value
AppQueueMaxSize	200
StackQueueMaxSize	100
SipSessionTableMaxSize	70000

> **Note:**   Because of the internal use of the queues by OCMS, the actual peak values for the queues in an overload situation may exceed the maximum values configured in the Overload Policy.

Overview of Overload Policy Architecture The Overload Policy receives collectors and actions from the Policy Manager (the lookup service for collectors, actions and policies). The Policy Manager sends notifications to all observers when available collectors, actions, or policies have changed.

Collectors Collectors notify policies when states change. The Overload Policy subscribes to state changes for the following collectors:

- Memory Usage
- Application Queue
- Network Queue
- SIP Session Table Usage

Overview of Overload Policy

The Overload Policy implements the following default actions, for which it enacts an overload action when high threshold values are reached and then reverts this action when usage sinks to the low threshold value:

- Do not accept new connections: This overload action occurs when a high warning threshold value has been met.
- Send 503 (*Service Unavailable*) response on initial requests: This overload action occurs when a high alarm threshold value has been met.
- Stop reading from all connections: This overload action occurs when a high critical threshold has been met.

Configuring High and Low Thresholds

The Overload Policy MBean enables you to configure the high and low values for the Warning, Alarm and Critical levels for Memory Usage, Application Queue, Network Queue, and SIP Session Table Usage. At each level, there may be one or even several actions to execute if usage exceeds the specified threshold. When the high value set for a threshold is met, the Overload Policy calls overload actions. This value is the threshold at which these actions start. If usage drops to the low value set for a threshold, then the Overload Policy stops the overload action. For example, if a high level is set to 90 and low level is set to 80, overload actions that start when usage reaches 90% and are then stopped when usage drops back to 80%.

Starting and Stopping the Overload Policy

The start and stop operations for the Overload Policy MBean enable you manually start and stop the Overload Policy. To automatically start the Overload Policy at the startup of OCMS, set the *Autostart* attribute to *true*.

Memory Usage

The Memory Monitor reports memory usage to the Overload Policy. The usage is reported as percent of total memory. For example, if the Memory Monitor reports a value of 85, it means that 85% of total memory is currently in use and that 15% of it is free.

Using the Overload Policy MBean, you configure the Warning, Alarm and Critical threshold levels for the Overload Policy's memory usage. At each level, there may be one or several actions to execute if memory usage exceeds the specified threshold. The MBean enables you to set high and low threshold values for each of these levels. The high value is the threshold at which to start executing overload actions. The low level marks the threshold at which to stop executing the action. For example, if high level is set to 90 and low level is set to 80, the actions start when memory usage reaches 90% and then stop when memory usage drops to 80% again.

Delaying a Memory Overload Action

High and low levels keep overload actions from starting and stopping repeatedly for small changes in memory usage. In addition to the threshold level that sets the start of the executing actions, you can also configure a delay time (in seconds) for memory overload actions using the *MemoryActionDelay* attribute. When the high threshold is exceeded, a scheduled timer fires after the specified delay time (in seconds). Memory overload actions are not be executed before the delay time has passed, and if memory usage drops below high threshold during the delay time, the timer is canceled and no actions executes.

Configure the *MemoryActionDelay* attribute by entering the delay (in seconds) from the instance when the memory threshold value has been exceeded to the instant the actions are executed. The execution stops if the memory drops below the threshold during the delay. The value set for the delay must be greater than 0. The default value is 60 seconds.

Table 9–9 lists the attributes that you configure to set the values for the high and low warning thresholds for memory usage.

Table 9–9   Warning High and Low Thresholds for Memory Usage

Attribute	Description
MemoryWarningHigh	The high value for a warning threshold that triggers an overload action to decline new connections. This is the default overload action. The range of values is 0-100. 0 disables the action. The default value is 95.
MemoryWarningLow	The low value for a warning threshold that triggers an end to the overload action. The range of values is 0-100. The default value is 90.
MemoryWarningActions	A comma-separated list of actions performed when a memory warning level has been reached. The default value is *oracle.sdp.networlayer.NetworkServiceImpl$StopAcceptAction*.

Table 9–10 lists the attributes that you configure to set the high and low alarm thresholds for memory usage.

Table 9–10   The Alarm HIgh and Low Thresholds for Memory Usage

Attribute	Description
MemoryAlarmHigh	The high value for an alarm threshold for memory usage that triggers an action to send a 503 response (*Service Unavailable*) on initial requests. This is the default overload action. The range of values is 0-100. 0 disables the action. The default value is 95.
MemoryAlarmLow	The low value for the alarm threshold for memory usage that triggers an end to the overload action. The range of values is 0-100. The default value is 90.
MemoryAlarmActions	A comma-separated list of actions performed when a memory alarm threshold level has been reached. The default value is *com.hotsip.jainsipimpl.javax.sip.context.SipContextImpl$Send503Action*.

Table 9–11 lists the attributes that you configure to set the high and low critical thresholds for memory usage.

Table 9–11   Critical High and Low Thresholds for Memory Usage

Attribute	Description
MemoryCriticalHigh	The high value for a critical threshold for memory usage that triggers an overload action to stop reading connections. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 98.
MemoryCriticalLow	The value that triggers an end to the overload action when the low threshold for memory usage has been reached. The range of values is 0-100. The default value is 90.
MemoryCriticalActions	A comma-separated list of action performed when the critical threshold of memory usage has been reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopReadAction*.

Application Queue

The Application Queue is the communication link between the network layer and the applications. An event is added to the Application Queue when network packages are framed and ready for application consumption after a session timer fires or other network related events occur. The network layer's EventNotifier reports Application Queue usage to the Overload Policy. This usage is reported as a percent of the total queue capacity. For example, if the EventNotifier reports a value of 85, it means that 85% of total queue capacity is currently used and that 15% of the queue is empty. The *AppQueMaxSize* attribute sets the capacity of the Application Queue. The default value is 200. This value must always be greater than zero.

The EventNotifier does not report every change in Application Queue usage to the Overload Policy: below 95%, every 5% change is reported (that is 0, 5, 10, 15 and so on). From 95% and above, every 1% change is reported (that is, 95, 96, 97 and so on). The Overload Policy MBean enables you to configure the Warning (Table 9–12), Alarm (Table 9–13), and Critical (Table 9–14) thresholds of the Application Queue.

Table 9–12 lists the attributes of the Overload Policy that enable you to set the high and low warning thresholds for the Application Queue usage.

Table 9–12   Warning High and Low Threshold and Actions for the Application Queue

Attribute	Value
AppQueueWarningHigh	The high threshold warning value for Application Queue usage that triggers an action to decline new connections. This is the default overload action. The range of values is 0-100. Selecting 0 disables the action. The default value is 70.

Table 9–12   (Cont.) Warning High and Low Threshold and Actions for the Application

Attribute	Value
AppQueueWarningLow	The low threshold warning value for Application Queue usage that triggers an end to the overload action. The range of values is 0-100. The default value is 50.
AppQueueWarningActions	A comma-separated list of actions performed when a warning threshold for Application Queue usage has been reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopAcceptAction*.

Table 9–13 lists the attributes of the Overload Policy that enable you to set the high and low alarm thresholds for the Application Queue usage.

Table 9–13   Alarm High and Low Threshold Actions for the Application Queue

Attribute	Description
AppQueueAlarmHigh	The high alarm threshold value for the Application Queue usage that triggers an overload action to send a 503 Response (*Service Unavailable*) to initial requests. (This is the default overload action.) The range of values is 0-100. 0 disables the action. The default value is 80.
AppQueueAlarmLow	The low alarm threshold value for Application Queue usage that triggers an end to the overload action. The range of values is 0-100. The default value is 60.
AppQueueAlarmActions	A comma-separated list of action performed when an alarm threshold is reached. The default value is *com.hotsip.jainsipimpl.javax.sip.context.SipContextImpl$Send503Action*.

Table 9–14 lists the attributes of the Overload Policy MBean that enable you to set the high and low critical thresholds for the Application Queue usage.

Table 9–14   Critical High and Low Threshold Actions for the Application Queue

Attribute	Description
AppQueueCriticalHigh	The high threshold value for Application Queue usage that triggers an overload action to stop reading connections. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 90.
AppQueueCriticalLow	The low threshold value for Application Queue usage that triggers an end to the overload action. The range of values is 0-100. The default value is 70.
AppQueueCriticalActions	A comma-separated list of actions performed when a critical level is reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopReadAction*.

Network Queue

The network layer deposits the incoming unframed data from the network into the Network Queue. The network layer's EventQueue reports the Network Queue usage to the Overload Policy. The usage is reported as a percent of the total queue capacity. For example, if the EventQueue reports a value of 85, it means that 85% of total queue capacity is currently in use and 15% of the queue is empty. The *StackQueueMaxSize* attribute sets the Network Queue capacity. The default value for this attribute is 100. The value must always be greater than zero.

The EventQueue does not report every change in queue usage to the Overload Policy. Below 95%, every 5% change is reported (that is, 0, 5, 10, 15 and so on). From 95% and above, every 1% change is reported (that is, 95, 96, 97 and so on). The Overload Policy MBean enables you to configure the Warning (Table 9–15), Alarm (Table 9–16), and Critical (Table 9–17) thresholds of the Network Queue.

Table 9–15 lists the attributes that you configure to set the high and low threshold values that trigger Warning actions.

Table 9–15   Warning High and Low Threshold Actions for the Network Queue

Attribute	Description
StackQueueWarningHigh	The high warning threshold value for Network Queue usage that triggers the overload action to decline new connections. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 70.
StackQueueWarningLow	The low warning threshold value that trigger end to the overload action. The range of values is 0-100. The default value is 50.
StackQueueWarningActions	A comma-separated list of actions performed when a warning threshold is reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopAcceptAction*.

Table 9–16 lists the attributes that you configure to set the high and low threshold values that trigger Alarm actions.

Table 9–16   Alarm High and Low Threshold Actions for the Network Queue

Attribute	Description
StackQueueAlarmHigh	The high alarm threshold for Network Queue usage that triggers an overload action to send a 503 (*Service Unavailable*) response to initial requests. This is the default action. The range of values is 0-100; 0 disables the action. The default value is 80.
StackQueueAlarmLow	The low alarm threshold for Network Queue usage that triggers an end to the overload action. The range of values is 0-100. The default value is 60.
StackAlarmQueueActions	A comma-separated list of actions performed when a threshold is reached. The default value is *com.hotsip.jainsipimpl.javax.sip.context.SipContexImpl$Send503Action*.

Table 9–17 lists the attributes that you configure to set the high and low values that trigger Critical actions.

Table 9–17   Critical High and Low Threshold Actions for the Network Queue

Attribute	Description
StackQueueCriticalHigh	The high critical threshold value for Network Queue usage that triggers an overload action to stop reading all connections. (This is the default overload action.) The range of values is 0-100. 0 disables the action. The default value is 90.
StackQueueCriticalLow	The low critical threshold value that triggers an end to the overload action. The range of values is 0-100. The default value is 70.
StackQueueCriticalActions	A comma-separated list of actions performed when a critical level is reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopReadAction*.

SIP Session Table Usage

The SIP servlet engine's Application Manager reports SIP session table usage to the Overload Policy. The usage is reported as a percent of total table capacity. For example, if the Application Manager reports a value of 85, it means that 85% of total table capacity is currently in use and 15% of it is free. The *SessionTableMaxSizeSip* attribute sets the SIP session table capacity. The default value of this attribute is 70000. The value must be greater than zero.

Not every change in table usage is reported back to overload policy. Below 95%, every 5% change is reported (that is, 0, 5, 10, 15 an so on). From 95% and above, every 1% change is reported (that is, 95, 96, 97 and so on). The Overload Policy MBean enables you to configure the Warning (Table 9–18), Alarm (Table 9–19), and Critical (Table 9–20) thresholds that trigger overload actions.

Table 9–18 lists the attributes that you configure to set the warning thresholds.

Table 9–18 Warning High and Low Threshold Actions for SIP Session Table Usage

Attribute	Description
SipSessionWarningHigh	The high warning threshold value for SIP session table usage that triggers an overload action to decline new connections. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 90.
SipSessionWarningLow	The low warning threshold that triggers an end to the overload action. The range of values is 0-100. The default value is 85.
SipSessionWarningActions	A comma-separated list of actions performed when a critical level is reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopAcceptAction*.

Table 9–19 lists the attributes that you configure to set the alarm level thresholds.

Table 9–19 Alarm High and Low Threshold Actions for SIP Session Table Usage

Attribute	Description
SipSessionAlarmHigh	The high alarm threshold of SIP session table usage that triggers an overload action to send a 503 Response (*Service Unavailable*) to initial requests. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 98.
SipSessionAlarmLow	The low alarm threshold of SIP session table usage that triggers an end to the overload action. The range of values is 0-100. The default value is 97.
SipSessionAlarmActions	A comma-separated list of actions performed when an alarm threshold is reached. The default value is *com.hotsip.jainsipimpl.javax.sip.context.SipContextImpl$Send503Action*.

Table 9–20 lists the attributes that you configure to set the critical thresholds.

Table 9–20 Critical Threshold Actions for SIP Session Table Usage

Attribute	Description
SipSessionCriticalHigh	The high critical threshold value for SIP session table usage that triggers an overload action to stop reading all connections. This is the default overload action. The range of values is 0-100; 0 disables the action. The default value is 100.
SipSessionCriticalLow	The low critical threshold for SIP session table usage that triggers an end to the overload action. The range of values is 0-100. The default value is 99.
SipSessionCriticalActions	A comma-separated list of actions performed when a critical level is reached. The default value is *oracle.sdp.networklayer.NetworkServiceImpl$StopReadAction*.

Deactivating the Overload Protection for System Tuning Technically, overload protection is always activated. When you tune the system, you can prevent overload actions from executing by doing either of the following:

■ Disabling all of the rules by setting all of the trigger values to zero (0).

■ Setting the *AppQueueMaxSize*, *StackQueueMaxSize*, and *SipSessionTableMaxSize* attributes to high numbers.

Memory Monitor

The Memory Monitor reports memory usage to the Overload Policy. The Memory Monitor polls the memory status from the runtime environment at either specified or random polling intervals. The Memory Monitor MBean includes attributes that enable you to select the type of polling interval and also the duration of the polling interval. Table 9–21 lists the attributes of the Memory Monitor MBean.

Table 9–21   Attributes of the Memory Monitor MBean

Attribute	Description
AutoStart	Select *true* to activate the Memory Monitor on startup of the SIP container.
PollingInterval	Enter the time, in seconds, between each interval that the Memory Monitor polls the memory status of the runtime environment. This attribute sets a fixed interval between polls. This value must be greater than five (5) seconds. The default value is 5.
RandomInterval	Select *true* to set the Memory Monitor to poll at a random intervals. The average length of these intervals will be the same as the value set for the *PollingInterval* attribute, but individual polls may differ by 50% from the fixed interval. For example, if the polling interval is set to 20 seconds, then a random interval may be pending between 10 and 30 seconds. The default setting is *false*.
MemoryMonitorStatus	The current status of the Memory Monitor. This value is read-only.
MemoryUsage	The current memory usage. This value is read-only.

Starting and Stopping the Memory Monitor

The start and stop operations enable you to manually start and stop the Memory Monitor.

SIP Cluster

The SIP Cluster MBean (SipCluster) enables you to override the default timeouts set for OC4J clusters. High Availability for OCMS is based on OC4J clusters. Because this framework is based on HTTP, the intervals set for replication are inappropriate for SIP, as they are on the order of seconds rather than milliseconds.

> **Caution:**   Because the values set for these timers can significantly impact high availability for OCMS, contact Oracle Support (http://www.oracle.com/) support before you change the values for this MBean. For most systems, the seeded values will suffice.

The SIPCluster MBean enables you to configure the following timers for state replication between peer nodes:

■ *RestoreTimeout* -- If a node cannot recognize the session ID of an incoming request, it requests a state replica from a peer. The interval set by the attribute reflects the time that the node waits for the requested state replica from another peer.

> **Note:** The default value for the restore timeout for OC4J clusters is 0, meaning that the peer would wait indefinitely for the state replica. Without the override provided by this attribute, the peer might block all of its active threads under some circumstances.

- *OwnedByTimeout* -- The time needed to pass ownership of the replicated data from one peer to another. This value sets the interval in which the node owning the state replica drops ownership and the requesting node confirms ownership. Once the requesting peer attains the data, its session replica is promoted to a "live" state.

> **Note:** The default value to pass ownership for OC4J clusters is -1, which instructs the peer not to wait for the replicated data. Because there is no interval between ownership, two peers can concurrently host "live" session replicas for a short period of time.

Configuring SIP Applications

This section describes the configuration enabled by the MBeans registered to the following SIP applications:

- Subscriber Data Services
- Presence (See "Overview of Presence")
- Proxy Registrar
- Application Router
- Aggregation Proxy

Subscriber Data Services

Subscriber Data Services (subscriberdataservices) is the parent application to all SIP applications that require authentication and security against the OCMS user repository. For example, the Application Router, Aggregation Proxy, and Proxy Registrar require Subscriber Data Services. In the case of the latter, the Location Service and the registrar component of the Proxy Registrar are dependent upon Subscriber Data Services. Subscriber Data Services also provides access to the TimesTen In-Memory database or Oracle Internet Directory (OID). For more information, see "Overview of Security". See also "Configuring Oracle Internet Directory as the User Repository" for information on using Oracle Internet Directory (OID), the LDAP data store used by Oracle WebCenter Suite, as the user provisioning repository for an OCMS deployment.

Account Security

Subscriber Data Services provides account security through the Account Lockout Service and Login Failure service MBean groups.

The Account Lockout group includes the following MBeans:

- AA Service

  The AA Service MBean is used by the Login Failure Service to lock accounts. This MBean depends on the MathFunction Model MBeans, which enable the AA Service MBean to calculate the next lock duration for an account based on the current number of failed login attempts.

> **Note:** Account locking persists when you restart OC4J.

Table 9–22 describes the attributes of the AA Service MBean.

Table 9–22   Attributes of the AA Service MBean

Attribute	Value
MathFunction	The type of math function used to calculate the next lock duration for an account based on the number of current failed login attempts. The math functions, which include `Linear`, `Exponential` and `Constant`, are packaged as ModelMBeans. The default value is *hotisp.math:service=MathFunction,name=Linear.*
DefaultLockDuration	The lock duration, in seconds, to use if no math functions are available. The default value is 600.
JNDIName	The JNDI Name of the AA Service. This value is read-only
SecurityServiceName	The JNDI Name bound to the service object that is used to unlock user accounts. This value is read-only.

Table 9–23   Subscriber Data Services MBeans

MBean	Tasks
ModelMBeanDeployer	The helper MBean used by the Subscriber Data Services application to deploy its Model MBeans.

- Constant (read-only)

  The constant function used by the MathFunction MBean to calculate the next lock duration for an account based on the number of current failed login attempts.

- Exponential

  The exponential math function used by the MathFunction MBean to calculate the next lock duration for an account based on the number of current failed login attempts.

- Linear

  The linear math function used by the MathFunction MBean to calculate the next lock duration for an account based on the number of current failed login attempts.

Of the Subscriber Data Services MBeans, only AA Service and Command Service can be configured.

CommandService

The operations of the CommandService MBean enable you to execute the equivalent of Sapphire Shell (Sash) commands which are used to provision OCMS users to the TimesTen In-Memory database. For example, to view a list of commands for account management:

1. Click the *Operations* tab. A list of `get` commands appears.

2. Click **help** for a returning a help String for a partial command. The parameters for the `help` operation appear.

3. Enter *account* in the *Value* field.

> **Tip:** Click **Use Multiline Editor** to expand the *Value* field to accommodate long command names.

4. Click **Invoke Operation**. The commands pertaining to account management appear (Figure 9–6). These commands match those retrieved by entering `help account` at the Sash prompt. For more information, see "Viewing Subcommands" in Chapter 5, "Provisioning Users and Applications".

Figure 9–6   Viewing Help for a Command

To view a list of all of the available commands (Figure 9–7), select *listAllCommands* from the *Operations* tab and then click **Invoke Operation** from the *listAllCommands* page that appears. For a description of Sash commands, see "Viewing Available Commands". For more information on user management through Sash, see "Creating a User".

Figure 9–7   Viewing Available Commands

The Command Service MBean is deployed within the presence application to enable user provisioning to the XDMS (XML Document Management Server). See Command Service (XDMS Provisioning).

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 9–24 describes the attributes of the Proxy Registrar.

Table 9–24   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 9–24   (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.

Application Router

When the OCMS SIP servlet container receives incoming requests that do not contain the application ID (`appId`) parameter, the container invokes the Application Router which inserts routing information that includes this parameter into the request's *ROUTE*. As a result, the container can respond to incoming `INVITE`, `MESSAGE`, `PUBLISH`, `REGISTER`, and `SUBSCRIBE` requests by invoking the appropriate applications in the correct order. As described in *Oracle Communication and Mobility Server Developer's Guide*, the `appId` parameter is an OCMS extension to JSR-116 used to route requests to applications.

Because requests generated by SIP clients other than Oracle Communicator typically do not contain the `appId` parameter in either their *ROUTE* or *REQUEST URI* headers, the presence of the `appId` parameter in an incoming request's *ROUTE* header ensures that the OCMS SIP servlet container correctly handles requests. The Application Router MBean (ocmsrouteloaderear, illustrated in Figure 9–8) enables you to configure the destinations for incoming `INVITE`, `MESSAGE`, `PUBLISH`, `REGISTER`, and `SUBSCRIBE` requests by defining the application routing sequence and other route-related criteria.

Figure 9–8   Setting the Application Routing for an INVITE Request
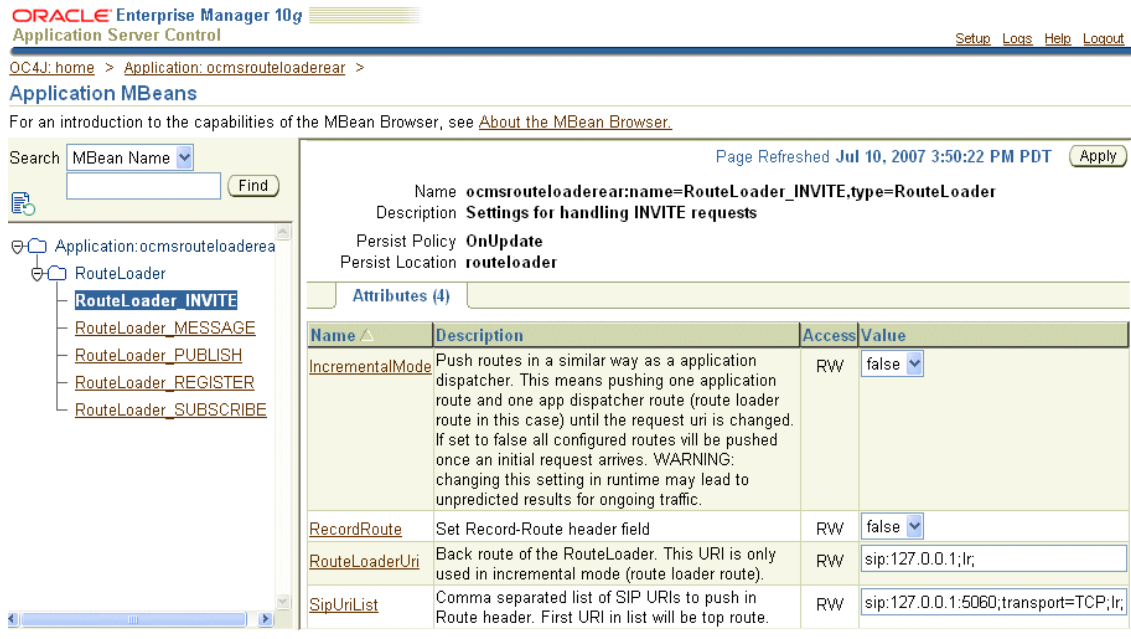
Table 9–25 describes the attributes that you configure for the routing of `INVITE`, `MESSAGE`, `PUBLISH`, `REGISTER`, and, `SUBSCRIBE` requests.

Table 9–25   Attributes of the Application Router

Attribute	Description
IncrementalMode	This boolean enables you to select the Application Router's execution mode: *true* for incremental, *false* for standard.
	■ Select *true* to set the incremental execution mode for the Application Router. For the incremental mode, the Application Router inserts the first (or top-most) URI configured in the *SIPURIList* attribute as well as the URI of the return route to the Application Router itself, which is defined as the value to the *RouteLoaderUri* attribute. When the request returns to the Application Router, the Application Router then checks if the Request URI has changed. If so, it proxies the request to this new URI. If the Request URI remains unchanged, then the Application router inserts the next URI defined in the *SIPURIList* into the request's ROUTE header and repeats the cycle.
	■ Select *false* to set the standard mode for the Application Router. In the standard mode, the Application Router inserts all of the routes defined in the *SipUriList* attribute in the request's ROUTE header. The request then follows the routes in the sequence that they are defined in the *SipUriList* attribute.

Table 9–25   (Cont.) Attributes of the Application Router

Attribute	Description
RecordRoute	Set to *true* to enable record routing.
RouteLoaderUri	The URI of the return route to the Application Router. This value must be the same URI as the SIP servlet container.
SipUriList	A comma-separated list of URIs, transport methods, and `appIds` of applications that the Application Router inserts in the ROUTE header of an incoming request.
	The default list, which routes requests to the Proxy Registrar, is defined as
	`sip:<SIP Container IP Address>:<SIP port>;transport=TCP;lr;appId=proxyregistrar`
	For example:
	`sip:144.25.174.189:5060;transport=TCP;lr;appId=proxyregistrar`
	Request routing is set according to the order of the applications listed in this attribute. For example, to call an application called *dialin* as the first destination for an `INVITE` request, insert the information for *dialin* as the first item on the list as follows:
	`sip:144.25.174.189:5060;transport=TCP;lr;appId=dialin,sip:144.25.174.189:5060;` `    transport=TCP;lr;appId=proxyregistrar`
	To ensure that request-related applications that you deploy to the OCMS SIP servlet container are invoked in response to incoming requests, you must add any application that you deploy to this list. The name of the application defined for the `appId` parameter (which is case-sensitive) can either be the full name of the deployed application, or an alias for the application. If the latter, the name must match the application alias set for the *ApplicationAliases* parameter of the SIP Servlet Container MBean. The Proxy Registrar must always be the last item listed in the *SIPUriList* attribute. See also "Deploying SIP Server Applications".
	For more information on the `appID` parameter, refer to *Oracle Communication and Mobility Server Developer's Guide*.

Overview of Presence

Presence represents the end-user's willingness and ability to receive calls. Client presence is often represented as a buddy 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 to extend presence service to end users. The application also enables service providers to base other services on presence information. The MBeans registered to the Presence application enable you to configure the presence service, which accepts, stores, and distributes presence information. See also "Presence Server" in Chapter 1, "An Overview of Oracle Communication and Mobility Server".

The Presence application MBeans enable you to manage the following:

- Presence Status Publication

- Presence Status Subscriptions

- Watcher-Info Support

- Presence XDMS Authorization of Subscriptions

- Privacy Filtering

- Presence Hard State

- Composition of Multiple Presence Sources

Presence Status Publication

A presentity can publish a PIDF (Presence Information Data Format) document containing presence state to the Presence Server.

Presence Status Subscriptions

The Presence server supports subscriptions to a user's status. The Presence Server notifies the user when the watcher (subscriber) is authorized to view the user's status. The Presence server also notifies all of the active, authorized watchers of the publication of a new presence document.

Watcher-Info Support

The Presence Server enables the user who is publishing presence information to subscribe to watcher-info events to receive information on all watchers currently subscribing to the user's presence information. The Presence Server also notifies users of changes in the watcher subscriptions, such as new or terminated subscriptions.

Presence XDMS Authorization of Subscriptions

Whenever a watcher subscribes to a user's presence, the Presence Server checks the authorization policy that the publisher has set to see if the subscriber has the required authorization.

If no matching rule can be found, the subscriber is put in a pending state and a watcher info notification is sent to the publisher. Usually, the publisher's client (User Agent) presents a pop-up box asking whether to accept or reject a new pending subscriber. The answer is added to the publisher's authorization policy document in the form of a rule for this subscriber. 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.

Privacy Filtering

A user can create privacy filtering rules that can be applied to a group or to the individual watcher, enabling different presence documents to be generated for different watchers. For example, a user can create rules which present a limited presence document for co-workers while friends receive the entire document.

Presence Hard State

The hard state feature enables a user to leave a document in the XDMS that notifies watchers when there are no other documents. In general, this feature is used for leaving an off-line note, such as "On Vacation".

Composition of Multiple Presence Sources

If a user has two or more clients (such as a PC and a mobile phone) both publishing presence documents, the Presence Server combines two or more documents into a unified document as dictated by a composition policy. The Presence server supports two different composition policies: a default policy and a policy that performs composition according to the OMA (Open Mobile Alliance) Presence enabler.

The default composition policy is a simple, but robust, algorithm. It adds `<dm:timestamp>` elements to the `<dm:person>` and `<dm:device>` elements if they are missing, and `<pidf:timestamp>` elements to the `<pidf:tuple>` elements if they are missing.

When the Presence Server creates the candidate document, it includes all `<pidf:tuple>` and `<dm:device>` elements from the source documents. It includes

only one `<dm:person>` element in the candidate document, and uses the latest published element based on the `<dm:timestamp>` element. All other `<dm:person>` elements are ignored.

Configuring Presence

Configuring the following MBeans enables Presence:

- Bus
- PackageManager
- Presence
- PresenceApplicationDeployer
- PresenceEventPackage
- PresenceWInfoEventPackage
- UA-ProfileEventPackage
- UserAgentFactoryService

Configuring XDMS The following MBeans enables you to configure the XDMS (XML Document Management Server):

- XCapConfig
- Command Service (XDMS Provisioning)

> **Note:**   If you change any attributes of the following MBeans, you must restart OCMS for these changes to take effect.
>
> - Presence
> - PresenceEventPackage
> - PresenceWInfoEventPackage
> - UAProfileEventPackage
> - XCAPConfig

Bus

The Bus MBean supports presence by setting the thread pool, the high and low watermarks for the job queues, and the duration that a job remains in the queue before notifications are dispatched. Table 9–26 describes the attributes of the Bus MBean.

Table 9–26   Attributes of the Bus MBean

Attribute	Value Type	Description
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 (if the current number of threads exceeds the value specified for *MinThreads*). The default value is 60.

Table 9–26 (Cont.) Attributes of the Bus MBean

Attribute	Value Type	Description
LogDuration	long	The duration, in seconds, that an event remains in the queue. 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.
MinThreads	int	The minimum number of threads held in the thread pool. If no threads are used, then the specified number of threads remains in an idle state, ready for upcoming jobs. The default value is 15.
MaxThreads	int	The maximum number of threads held in the thread pool. When the specified number of threads are occupied with jobs, subsequent jobs are placed in a queue and are dealt with as the threads become available. The default value is 10.

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 subscribers 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 for the PresenceEventPackage, WatcherinfoPackage, and UA-ProfileEventPackage Means. Table 9–27 describes the attributes of the PackageManger MBean.

Table 9–27 Attributes of the EventPackages MBean

Attribute	Description
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*.

Table 9–27   (Cont.) Attributes of the EventPackages MBean

Attribute	Description
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.

Presence

The Presence MBean controls how the Presence Server interacts with presentities, Publish User Agents (PUAs) that provide presence information to presence services. The attributes (described in Table 9–28) include those for setting the composition policy for creating a unified document when a user publishes presence documents from two or more clients, as well as setting the blocking, filtering, and presence hard state.

Table 9–28   Attributes of the Presence MBean

Attribute	Description/Value
CompositionPolicyFilename	The filename of the composition policy document. Values include `compose.xslt`, for the OCMS 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. For more information about this, see *Chapter 3.2.1: Subscription Handling* in the IETF SIMPLE draft for presence rules (`http://www.ietf.org/internet-drafts/draft-ietf-simple-presence-rules-04.txt`).
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. If the number of users is large, Oracle recommends that you store the presence documents on file. Valid values: ■ file ■ memory The default value is *memory*.

Table 9–28 (Cont.) Attributes of the Presence MBean

Attribute	Description/Value
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	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*. Unauthenticated users are blocked when no rule is found. If the URI contains a domain name instead of an IP address, then you must configure the DNS Server. 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*.
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 user's presence will always see everything that has been published for the presentity.
TransformerFactory	The name of the `TrasformerFactory` class. The default value is `net.sf.saxon.TransformerFactoryImpl.`

PresenceEventPackage

Table 9–29 describes 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 to set the interval of the expiry of a publication or a subscription that is accepted by the Presence Server. A client states when its publication or subscription expires. If a client sends an expiry time that is lower than the configured `minexpires` time, the server returns a 423 (*Subscription Too Brief*) response. If a client sends an expires time that is higher than the configured `maxexpires` time, the server returns the `maxexpires` time in the response. 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 9–29   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 long*. 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.
NotifierMaxExpires	The maximum time, in seconds, for a SUBSCRIBE to expire. The default is 3600.
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`.

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 subscriptions of the Watcher Information Event Package. As a result, the subscriber learns of changes in the monitored resources subscriptions.

The PresenceWInfoEventPackage MBean (described in Table 9–30) sets the subscription state information for the Watcher Information Event Package.

Table 9–30   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`.

Table 9–30   (Cont.) Attributes of the WatcherinfoEventPackage

Attribute	Description/Value
NotifierMaxDocumentSize	The maximum document size for SUBSCRIBE.
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`.

UA-ProfileEventPackage

Table 9–31 describes the attributes of the UA-ProfileEventPackage MBean.

Table 9–31   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*.
Notifier	The name of the `Notifier` class. The default value is: `oracle.sdp.presenceeventpackage.PresenceSubscriptionControl`
NotifierMaxDocumentSize	The maximum document size for a SUBSCRIBE.
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`

UserAgentFactoryService

The UserAgentFactoryService MBean sets the commands for user agent factory service. The Presence Server uses the user agent factory to subscribe to changes in XML documents stored in the XDMS for presence.

Table 9–32   Attributes of the UserAgentFactoryService MBean

Attribute Name	Description/Value
DNSNames	A comma-separated list of DNS (Domain Name System) IP addresses used by the user agent.
IpAddress	The IP address for the user agent client; use the empty string (the default setting) for the default network interface on the current system.
PreferredTransport	The preferred transport protocol that enables communication between the Presence Server and the XDMS. The default value is TCP. Valid values are TCP and UDP.
Port	The IP port for the user agent client. The default value is 5070.

Command Service (XDMS Provisioning)

The Command Service MBean enables user provisioning to the XDMS. For more information see "CommandService".

XCapConfig

The XCapConfig MBean controls the configuration of the XDMS, the repository of the XCAP (Extensible Markup Language Configuration Access Protocol) documents containing user presence rules (pres-rules) and hard state information. The XCapConfig MBean settings can be ignored if the XDMS is external to OCMS.

Table 9–33   Attributes of the XCapConfig 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 XDMS document. Although Oracle recommends a default maximum size per XDMS document of 1 MB (1000 contacts at about 1 KB each), 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 file URL of the XCAP document files storage root. For example: *file:/var/tmp/xcaproot/*
PidfManipulationAuid	The ID 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*. Unauthenticated users are blocked when no rule is found. If the URI contains a domain name instead of an IP address, then you must configure the DNS Server.
	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*.

Table 9–33   (Cont.) Attributes of the XCapConfig MBean

Attribute Name	Description/Value
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://127.0.0.1:8080/services.* The URL defined in this parameter gives clients the location of the content server (which can be on a separate server from the XDMS). The XDMS places this URL in the *Content-Type* header of its outgoing NOTIFY messages. For example, the *Content-Type* header in the following NOTIFY message from the XDMS to the Presence Server notes that the body of the pres-rules document is stored externally and also includes instructions within the URL for retrieving the document.

```
CSeq: 1 NOTIFY
From: <sip:bob_0@144.22.3.45>;tag=66910936-0e31-41b2-abac-10d7616d04ef
To: <sip:bob_0@144.22.3.45>;tag=ffa3e97bd77f91e6ca727fbf48a5678b
Content-Type:
message/external-body;URL="http://127.0.0.1:8888/contentserver/pres-rul
es/users/bob_0@144.22.3.45/presrules";access-type="URL"
...
Event:
ua-profile;document="pres-rules/users/sip:bob_
0@144.22.3.45/presrules";profile
-type=application;auid="pres-rules"
```

Attribute Name	Description/Value
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 Aggregation Proxy. If this attribute is set to *true*, then any incoming XCAP request that lacks an asserted identity is denied access.

Configuring Presence Web Services

OCMS 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 can build this client using their knowledge of Web Services.

The Presence Web Services application, which is deployed as a child application of the Presence application, contains the following MBeans that enable you to configure a Web Services deployment server:

- Bus
- PackageManager
  - PresenceEventPackage
  - PresenceWInfoEventPackage
  - UA-ProfileEventPackage
- Presence
- UserAgentFactoryService
- XCapConfig

The Presence Web Services application also includes the PresenceSupplierWebService and PresenceConsumerWebService MBeans, which contain attributes for managing presence publication and watcher subscriptions enabled through the OCMS implementation of Presence Consumer and Presence Supplier interfaces. For more information on the OCMS implementation of Parlay X Web Service and support of these interfaces, refer to *Oracle Communication and Mobility Server Developer's Guide*.

PresenceWebServiceDeployer

Starts the JMX framework for the Presence Web Services application and deploys all of its Model MBeans. The operations of the PresenceWebServiceDeployer MBean enable you to retrieve information of the objects exposed by the Presence Web Service to this MBean.

Table 9–34   Operations of the PresenceWebServiceDeployer MBean

Operation	Description
getManagedObjectNames	Returns a String array containing the object names of the deployed application.
getMBeanInfo	Returns the meta-data for the deployed MBean.
getMBeanInfo (locale)	Returns the localized meta-data for the deployed Mbean.

PresenceSupplierWebService

The PresenceSupplierWebService MBean (described in Table 9–35) enables you to manage the presence data published to watchers.

Table 9–35   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*. Unauthenticated users are blocked when no rule is found. If the URI contains a domain name instead of an IP address, then you must configure the DNS Server. 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*.
PublicXCAPRootUrl	The URL to the public XDMS root, entered as *http://<your.xdms:domain.com>/services/*. For example, enter *http://127.0.0.1:8080/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.

Table 9–35   (Cont.) Attributes of the PresenceSupplierWebService MBean

Attributes	Description
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>;lr;transport=UDP
	You can also enter the default port (5060) in this address. For example, enter *sip:127.0.0.1:5060;lr;transport=UDP*. The shortest format for entering this address is *sip:127.0.0.1;lr*.
	If you do not define this attribute, then no outbound proxy will be used.

PresenceConsumerWebService

The PresenceConsumerWebService MBean (described in Table 9–36) enables you to set the duration of watcher subscriptions.

Table 9–36   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.
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>;lr;transport=UDP
	You can also enter the default port (5060) in this address. For example, enter *sip:127.0.0.1:5060;lr;transport=UDP*. The shortest format for entering this address is *sip:127.0.0.1;lr*.
	If you do not define this attribute, then no outbound proxy will be used.

Aggregation Proxy

The Aggregation Proxy is a server-side entry point for OMA clients that authenticates any XCAP traffic and Web Service calls (which are conducted through HTTP, not SIP) by providing identity assertion. This component acts as the gatekeeper for the trusted domain that houses the Presence Server and the XDMS.

The Parlay X Web Service operates within a trusted domain where the Aggregation Proxy authorizes the user of the Web Service. It authenticates XCAP traffic and Web Service calls emanating from a Parlay X client by inserting identity headers that identify the user of the Web Services. The Aggregation Proxy then proxies this traffic (which is sent over HTTP) to the Parlay X Web Service and XDMS.

> **Note:**   The Aggregation Proxy uses the OCMS Logging modules.

The attributes of the Aggregation Proxy MBean (Table 9–37) 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 9–37   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	Hostname 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.
JAASLogingContext	The `name` for the JAAS (Java Authentication and Authorization Service) `javax.security.auth.login.LoginContext`.
JAASRoles	A comma-separated list of JAAS roles for authentication.
PresenceConsumerEndpoint	The path to the endpoint of the Presence Consumer Web Service. The methods of the Presence Consumer interface enable watchers to obtain presence data. For more information, refer to *Oracle Communication and Mobility Server Developer's Guide*.
PresenceSupplierEndpoint	The path to the endpoint of the PresenceSupplier Web Service. The methods of the Presence Supplier Interface enable presentities to provide presence manage the data accessed by watchers. For more information, refer to *Oracle Communication and Mobility Server Developer's Guide*.
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.
WebServiceHost	The host name of the Web Services deployment server to which the Aggregation proxies requests.
WebServicePort	The port of the Web Services deployment server to which the Aggregation proxies requests.
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.

Securing the XDMS with the Aggregation Proxy

Secure the XDMS by deploying it behind the Aggregation Proxy. Access to the XDMS should be restricted only to the Aggregation Proxy and the Presence Server. In addition, securing the XDMS requires that you configure the Presence Sever application's XCapConfig MBean, the Aggregation Proxy and the Oracle Communicator as follows:

- Deny access to any incoming XCAP request that lacks an asserted identity header by setting the value of the *RequiredAssertedIdentity* attribute of Presence Server's

XCAPConfig MBean to *true*. Setting this attribute to *true* requires authentication of all XCAP by the Aggregation Proxy.

- Set the appropriate XDMS-related values for the *XCAPHost*, *XCAPPort*, *XCAPRoot*, *ContentHost*, *ContentPort* and *ContentRoot* attributes of the Aggregation Proxy MBean.

- Configure the Oracle Communicator's XDMS settings in `customize.xml` to point to the Aggregation Proxy -- not to the XDMS -- by defining the `<RootContext>` element as `aggregationproxy`, the context root of the Aggregation Proxy and by setting the `<host>` and `<port>` elements to the host of the Aggregation Proxy and the HTTPS port on that host, such as 443. See also "Enabling Presence" in "Configuring Oracle Communicator".

The Aggregation Proxy must be deployed as a child application of Subscriber Data Services. You can bind to the `default-web-site` for HTTP. To enable HTTP over SSL, you must configure the OC4J Container on which the Aggregation Proxy executes to provide HTTPS. Refer to *Oracle Containers for J2EE Security Guide* for instructions on configuring HTTPS. To enable access to the Aggregation Proxy over HTTPS, bind the Aggregation Proxy with the `secure-web-site`. Ensure that the Presence Server binds with the `default-web-site` if it resides on the same server with the Aggregation Proxy. Because the Presence Server resides in the `presence.ear` file, all of the HTTP servlets in that EAR file must bind to `default-web-site`.

> **Note:** Oracle Communication and Mobility Server Version 10.1.3.2 does not support direct access to the XDMS by HTTPS.

Viewing Log Files

Log files are named after the component loggers. For example, the log for the *traffic* component is called the *traffic.log*. OCMS writes logs for the *traffic*, *system*, *statistics*, *config* and *badsmg* components to the log files located at the `ORACLE_HOME/j2ee/home/log/sdp` directory in standalone OC4J deployments. OCMS writes these logs to `ORACLE_HOME/J2EE/ocms/log/sdp` in enterprise deployments.

> **Note:** Debug messages are written to the system directory, which is also located in the *sdp* directory.

You can view these log files using any text editor. See also "Configuring the Logging System" and "SIP Servlet Container Logging".

Deploying SIP Server Applications

You can deploy, undeploy and redeploy SIP applications to OC4J using either the wizard provided in Application Server Control Console or the `admin_client.jar` command-line tool. This section gives a brief overview of both of these options through the following topics:

- "Deploying, Undeploying, and Redeploying SIP Servlet Applications with Application Server Control"

- "Deploying, Undeploying, and Redeploying an Application Using the admin_client.jar Utility"

For more information, refer to *Oracle Containers for J2EE Deployment Guide*.

> **Note:** SIP applications can only be deployed to OC4J if they are
> packaged into a J2EE-compliant EAR file. For more information, see
> "Packaging and Deploying Applications".

Deploying, Undeploying, and Redeploying SIP Servlet Applications with Application Server Control

The Application Sever Control Console provides a wizard that steps you through
deploying, undeploying, and redploying SIP applications.

> **Tip:** Use a firewall to block all incoming SIP traffic until all of the
> applications have been fully deployed and the server started.

You can filter SIP traffic using a shell script, such as the following
Linux script, `blockport.sh`, which uses the iptables tool.

```
#!/bin/bash

if [ $# != 1 ]
then
  echo "blockport.sh <port>"
  exit
fi

iptables -A INPUT -p tcp -m tcp --dport $1 -j DROP
iptables -A INPUT -p udp -m udp --dport $1 -j DROP
service iptables save

echo "Port "$1" blocked."
```

Likewise, you can use a shell script to enable the flow of SIP traffic
once the server is running and all applications have been fully
deployed. The following Linux script, `unblockport.sh`, is an
example of script that enables SIP traffic:

```
#!/bin/bash

if [ $# != 1 ]
then
  echo "unblockport.sh <port>"
  exit
fi

iptables -D INPUT -p tcp -m tcp --dport $1 -j DROP
iptables -D INPUT -p udp -m udp --dport $1 -j DROP
service iptables save

echo "Port "$1" unblocked."
```

For more information, see *Deploying with Application Server Control Console* in *Oracle
Containers for J2EE Deployment Guide*.

Deploying an Application using the Deployment Wizard

The **Deploy** button on the *Applications* page invokes the deployment wizard which
guides you through the deployment process through the following pages:

■ The *Select Archive* page (Figure 9–9) is the first page of the wizard. To complete this page, point OC4J to the location of the EAR (Enterprise Archive) file containing the SIP application. This page also enables you to select the option to create or apply a deployment plan, a client-side aggregation of all of the configuration data needed to deploy an archive into OC4J. If you use an existing deployment plan, you enter its location. If you opt for new deployment plan, select the **Automatically Create a New Deployment Plan** option. Refer to "Packaging and Deploying Applications" for information on EAR file structure and how to package a SIP application for deployment.

> **Tip:** The wizard automatically creates a new deployment plan if you do not enter the location of an existing plan.

Figure 9–9   Deploying an Application: Entering the Archive Location
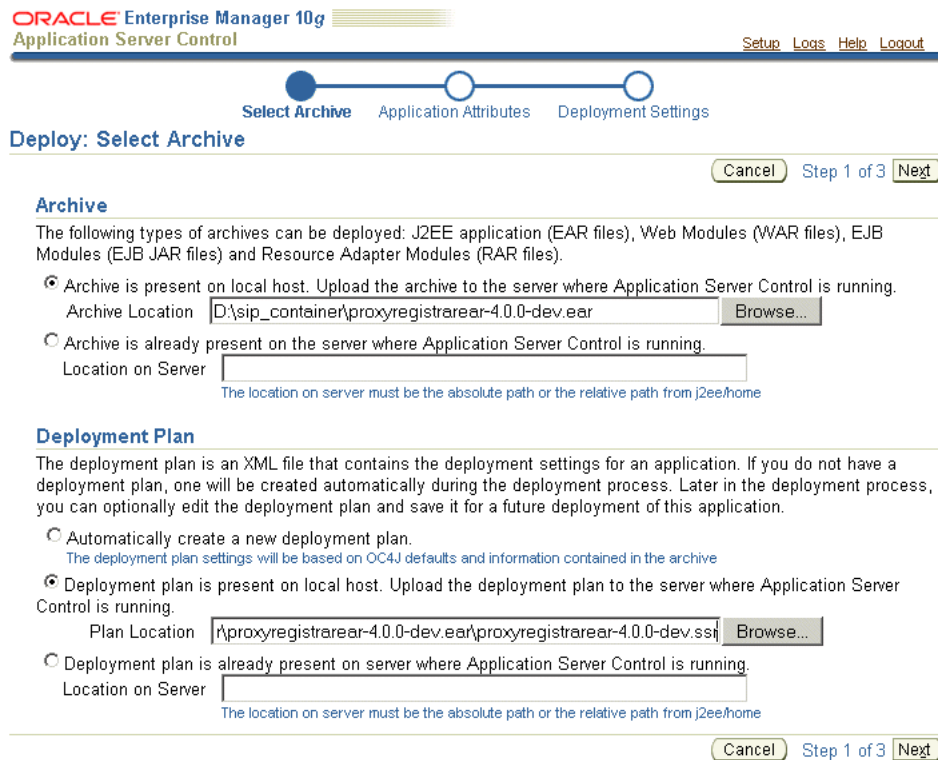
■ Clicking **Next** invokes the *Application Attributes* page (Figure 9–10). This page enables you to enter the application name and select the parent application. The application name cannot contain spaces.

Select Subscriber Data Services as the parent application for OCMS applications requiring authentication.

> **Note:** The SIP application becomes a child of the default application if you do not specify a parent application.

The *Application Attributes* page also enables you to set the binding of a Web application to a Web site by specifying the name portion of the `name-web-site.xml` configuration file that defines the Web site. A Web

application deployed as part of a J2EE application must be bound to the Web site through which it is accessed.

The Web module context root, which will be appended to the URL used to access the application through a Web browser, is also set as part of the process to enable Web access. This value is typically read from the `application.xml` deployment descriptor packaged with the application.

Figure 9–10   Deploying an Application: Entering the Application Name and Parent Application

- The *Deployment Settings* page (Figure 9–11) provides a tasks that enable you to edit the deployment plan.

Figure 9–11   Deploying an Application: Configuring the Deployment Settings

Complete deployment tasks as needed and then click **Deploy**. The confirmation page appears (Figure 9–12).

Figure 9–12   Confirming the Deployment

> **Note:** To ensure that the OMCS SIP servlet container responds appropriately to incoming requests, you must add any deployed application that processes requests to the Application Router's *SIPUriList* attribute. Configuring this attribute ensures that applications are added to the *ROUTE* header of incoming INVITE, MESSAGE, PUBLISH, REGISTER, or SUBSCRIBE requests from non-OCMS (that is, non-OCMS Communicator) SIP clients. The Proxy Registrar must always be the last item listed in the *SIPUriList* attribute.

Undeploying an Application Using the Deployment Wizard

You can remove (undeploy) an application by first selecting it and then by clicking the **Undeploy** button. When you undeploy an application, you likewise undeploy the MBeans registered to the application.

Likewise, if you undeploy a parent application, its the child applications are also undeployed. As a result, the parent application and all related applications must be redeployed. See *Oracle Containers for J2EE Deployment Guide* for information on when to restart OC4J when undeploying an application.

Redeploying an Application Using the Deployment Wizard

The **Redeploy** button enables you undeploy an application without restarting OC4J. Redeploying a SIP application packaged within an EAR file prompts OC4J to undeploy the previous instance; you do not have to first select the application and then click **Undeploy**.

Like deploying an application, the wizard prompts you through a three-step process for redeploying an application in which you point OC4J to the EAR file, select or create a deployment plan, select the parent application and Web bindings, and complete deployment descriptor configuration tasks.

Deploying, Undeploying, and Redeploying an Application Using the admin_client.jar Utility

The admin_client.jar command-line utility used to perform deployment-related operations on active OC4J instances in an Oracle Application Server clustered environment as well as on standalone OC4J servers.

The admin_client.jar utility is installed by default in ORACLE_HOME/j2ee/home in an OC4J instance. OC4J must be started before this utility can be used.

Deploying an Application Using admin_client.jar

To deploy an EAR, use the -deploy command with the EAR-specific context as follows:

```
java -jar admin_client.jar uri adminId adminPassword -deploy -file
path/filename -deploymentName appName [-bindAllWebApps [webSiteName]]
[-targetPath path] [-parent appName] [-deploymentDirectory path]
-enableIIOP [-iiopClientJar path/filename]
```

Undeploying an Application Using admin_client.jar

To undeploy an application:

```
java -jar admin_client.jar uri adminId adminPassword -undeploy appName
```

Redploying an Application Using admin_client.jar

To redeploy a previously deployed archive, use the `-redeploy` command and with the following syntax:

```
java -jar admin_client.jar uri adminId adminPassword -redeploy -file
path/filename -deploymentName appName [-keepSettings] [-sequential]
```

Refer to *Deploying with the admin_client.jar Utility* in *Oracle Containers for J2EE Deployment Guide* for more information on the `-redeploy` command subswitches.

Managing OCMS through JBoss Application Server

You can install OCMS on top of JBoss Application Server (Version 4.0.5) by Red Hat Inc.

> **Note:** Oracle High Availability is not supported when OCMS is installed on top the JBoss Application Server.

Starting and Stopping OCMS through JBoss Application Server

This section describes how to start and stop OCMS on both Windows and Linux operating systems.

Starting and Stopping OCMS on Windows

To start OCMS on Windows:

From **Programs**, select **Oracle Communication and Mobility Server** and then select **Start OCMS**.

To stop OCMS:

From **Programs**, select **Oracle Communication and Mobility Server** and then select **Stop OCMS**.

Starting and Stopping OCMS on Linux

To start OCMS on Linux systems:

Enter `JBOSS_HOME/sdp/bin/startocms`

To stop OCMS:

Enter `JBOSS_HOME/sdp/bin/stopocms`

Managing OCMS MBeans through JBoss Application Server

Using The JMX Console, you can configure the OCMS MBeans and invoke the MBean operations. The MBean attributes exposed through this console are the same as those exposed through OC4J.

Once OCMS starts and initializes JBoss Application Server, you can access OCMS MBeans and applications as follows:

1. In a browser, enter host and port for the JBoss Application Server. For example, enter *http://localhost:8080/*. The default page appears (Figure 9–13).

Figure 9–13   The Default Page of the JBoss Application Server

2.   Access the JMX Agent View by clicking either **JMX Console** or **JBoss Web Console**. If you select the later, expand *System* in the navigation tree and then select *JMX MBeans* (Figure 9–14).

> **Tip:**   Use the ObjectName Filter (illustrated in Figure 9–14) to locate specific MBeans. For example, to locate the SIP Servlet Container-related Mbeans described in "Managing OCMS MBeans", enter **:type=SipContainer,** and then click **ApplyFilter**.

Figure 9–14   OCMS MBeans Accessed through the JMX Console
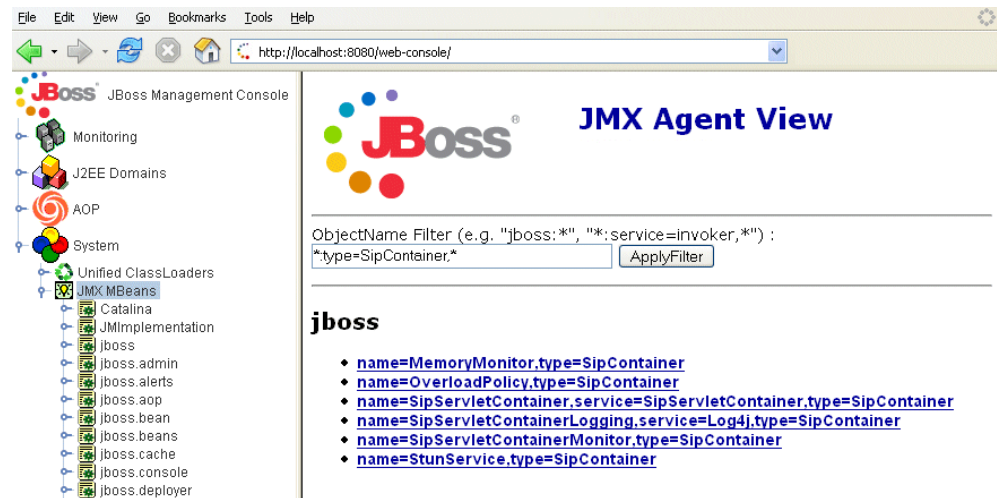
3.   Click the link of an MBean or application to access the attributes. For example, to access the attributes and operations of the "SIP Servlet Container" MBean, click the *name=SipServletContainer,service=SipServletContainer,type=SipContainer* link. The *JMX Bean View* page appears (Figure 9–15), displaying the attributes and operations.

Figure 9–15   Viewing Attributes and Operations

10

Configuring the Logging System

This chapter describes the Apache log4j-based logging framework used by OCMS. This chapter includes the following sections:

- "Overview of log4j Logging in OCMS"
- "Defining Rolling File Appenders for Core Components"
- "Setting the Log Levels for Core Components"
- "Exposing a Component's Log Levels through the SIP Servlet Container Logging MBean"
- "Viewing Application Log Files"

Overview of log4j Logging in OCMS

The global `log4j.xml` file enables you to modify the behavior of the *traffic*, *system*, *config*, *badmsg* and *statstics* component loggers. Changes made to this file persist; changes that you make to the logging file through the SIP Servlet Container Logging MBean are temporary.

You configure the log4j logging system by editing the global `log4j.xml` properties file, located within `ORACLE_HOME/j2ee/home/lib/ext` in Oracle Containers for J2EE (OC4J) or in the `$JBOSS_HOME/server/$jboss_server_name/conf` directory in the JBoss Application Server. Upon initialization of the application server, the log4j logging system configures itself by reading the global `log4j.xml` properties file.

> **Note:** The CUSTOMER_FILE_LOG in Example 10–1 is the *system* logger.

Defining Rolling File Appenders for Core Components

You configure the rolling file appenders used for component loggers by defining the global `log4j.xml` file (illustrated in Example 10–1). The default configuration of `log4j.xml` defines rolling file appenders for each of these loggers.

Example 10–1   The Default Rolling File Appenders in log4j.xml

```
<!-- A size based rolling appender -->
  <appender name="CUSTOMER_FILE_LOG"
    class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${oracle.j2ee.home}/log/sdp/system/system.log" />
    <param name="Append" value="true" />
```

```xml
      <param name="Encoding" value="UTF-8" />
      <param name="MaxFileSize" value="100000KB" />
      <param name="MaxBackupIndex" value="10" />

      <layout class="org.apache.log4j.PatternLayout">
        <!-- The default pattern: Date Priority [Category] Message\n -->
        <param name="ConversionPattern"
          value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} %-5r %-5p [%c{1}] (%t:%x) %m%n" />
      </layout>
  </appender>

  <!-- A size based rolling appender -->
  <appender name="TRAFFIC_FILE_LOG"
    class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${oracle.j2ee.home}/log/sdp/traffic/traffic.log" />
    <param name="Append" value="true" />
    <param name="Encoding" value="UTF-8" />
    <param name="MaxFileSize" value="100000KB" />
    <param name="MaxBackupIndex" value="10" />

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] Message\n -->
      <param name="ConversionPattern"
        value="%d{ISO8601} %-5p [%c{1}] (%t:%x) %m%n" />
    </layout>
  </appender>

  <!-- A size based rolling appender -->
  <appender name="BADMSG_FILE_LOG"
    class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${oracle.j2ee.home}/log/sdp/badmsg/badmsg.log" />
    <param name="Append" value="true" />
    <param name="Encoding" value="UTF-8" />
    <param name="MaxFileSize" value="100000KB" />
    <param name="MaxBackupIndex" value="10" />

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] Message\n -->
      <param name="ConversionPattern"
        value="%d{ISO8601} %-5p [%c{1}] (%t:%x) %m%n" />
    </layout>
  </appender>

  <!-- A size based rolling appender -->
  <appender name="CONFIG_FILE_LOG" class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${oracle.j2ee.home}/log/sdp/config/config.log" />
    <param name="Append" value="true"/>
    <param name="Encoding" value="UTF-8" />
<param name="MaxFileSize" value="500KB"/>
<param name="MaxBackupIndex" value="2"/>

    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date [Category] Message\n -->
      <param name="ConversionPattern"
        value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} [%c{1}] %m%n" />
    </layout>
  </appender>

  <!-- A size based rolling appender -->
  <appender name="STATISTICS_FILE_LOG"
```

```
class="org.apache.log4j.RollingFileAppender">
    <param name="File"
value="${oracle.j2ee.home}/log/sdp/statistics/statistics.log" />
    <param name="Append" value="true" />
    <param name="Encoding" value="UTF-8" />
    <param name="MaxFileSize" value="100000KB" />
    <param name="MaxBackupIndex" value="10" />

    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} %m%n" />
    </layout>
  </appender>

  <appender name="STATISTICS_FORMAT_FILE_LOG"
class="org.apache.log4j.RollingFileAppender">
    <param name="File"
value="${oracle.j2ee.home}/log/sdp/statistics/statistics.log" />
    <param name="Append" value="true" />
    <param name="Encoding" value="UTF-8" />
    <param name="MaxFileSize" value="100000KB" />
    <param name="MaxBackupIndex" value="10" />

    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern"
        value="%m%n" />
    </layout>
  </appender>

  <!-- A size based rolling appender -->
  <appender name="EVENTLOGGER_FILE_SIZE"
    class="org.apache.log4j.RollingFileAppender">
    <param name="File" value="${oracle.j2ee.home}/log/sdp/events.log" />
    <param name="Append" value="true" />
    <param name="Encoding" value="UTF-8" />
    <param name="MaxFileSize" value="100000KB" />
    <param name="MaxBackupIndex" value="10" />
    <layout class="oracle.sdp.eventlogger.BasicLayout"></layout>
  </appender>

  <!-- ============================= -->
  <!-- Append messages to the console -->
  <!-- ============================= -->
  <appender name="CONSOLE" class="org.apache.log4j.ConsoleAppender">
    <param name="Threshold" value="ERROR" />
    <param name="Encoding" value="UTF-8" />
    <layout class="org.apache.log4j.PatternLayout">
      <!-- The default pattern: Date Priority [Category] Message\n -->
      <param name="ConversionPattern"
        value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} %-5p [%c{1}] %m%n" />
    </layout>
  </appender>

  <appender name="SYSTEM_SOCKET"
    class="org.apache.log4j.net.SocketAppender">
    <param name="RemoteHost" value="127.0.0.1" />
    <param name="Port" value="4477" />
    <param name="Threshold" value="DEBUG" />
  </appender>
```

```
<!-- Syslog events -->
<appender name="TRAFFIC_SYSLOG"
  class="org.apache.log4j.net.SyslogAppender">
  <param name="Threshold" value="INFO" />
  <param name="Facility" value="LOCAL1" />
  <!-- param name="FacilityPrinting" value="false"/ -->
  <param name="SyslogHost" value="127.0.0.1" />
  <layout class="org.apache.log4j.PatternLayout">
    <!-- The default pattern: Date Priority [Category] Message\n -->
    <param name="ConversionPattern"
      value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} %-5r %-5p [%c] (%t:%x) %m%n" />
  </layout>
</appender>

<appender name="SYSTEM_SYSLOG"
  class="org.apache.log4j.net.SyslogAppender">
  <param name="Threshold" value="ERROR" />
  <param name="Facility" value="LOCAL0" />
  <!-- param name="FacilityPrinting" value="false"/ -->
  <param name="SyslogHost" value="127.0.0.1" />
  <layout class="org.apache.log4j.PatternLayout">
    <!-- The default pattern: Date Priority [Category] Message\n -->
    <param name="ConversionPattern"
      value="%d{yyyy-MM-dd HH:mm:ss,SSSZ} %-5r %-5p [%c] (%t:%x) %m%n" />
  </layout>
</appender>

<appender name="EVENTLOGGER_SYSLOG"
  class="org.apache.log4j.net.SyslogAppender">
  <param name="Threshold" value="ALL" />
  <param name="Facility" value="LOCAL2" />
  <!-- param name="FacilityPrinting" value="false"/ -->
  <param name="SyslogHost" value="127.0.0.1" />
  <layout class="oracle.sdp.eventlogger.BasicLayout"></layout>
</appender>
```

Logging Output

The output of the components is written to the log files located within the log/sdp directory of the OCMS OC4J instance's J2EE home (or through the $JBOSS_ HOME/server/$jboss_server_name/log/sdp directory of the JBoss Application Server). In standalone OC4J, the log files are located within ORACLE_ HOME/j2ee/home; in Oracle Application Server, the log files are located at ORACLE_ HOME/j2ee/OCMS. These components write output using the pattern of

```
Date MS Priority
Category (Thread:NDC) Message
```

For more information, see "Viewing Log Files".

The log files are automatically backed up by log4j's Rolling File Appender implementation. A minimum number of backups is maintained (up to the MaxBackupIndex), each backup having a maximum file size as specified in the MaxFileSize log4j configuration parameter.

Setting the Log Levels for Core Components

You can set the logging level of components dynamically through a JMX console (such as the Oracle Application Server System MBean browser), or statically by editing the global `log4j.xml` file. On Oracle Application Server, the log4j configuration changes that are made to the `log4j.xml` file are picked up automatically every 60 seconds and do not require a server reboot. On a standalone OC4J instance, the server must be restarted for the configuration changes to take effect.

Setting the Log Levels through the Oracle Application Server Control MBean Browser

The log level for the components can be modified at runtime by modifying the attributes of the SIP Servlet Container Logging MBean using either Application Server Control (Figure 10–1) as described in "Accessing SIP Servlet Container MBeans", or through the JBoss application server's JMX console. The attributes of the SIP Servlet Container Logging MBean enable you to set the log levels for the components to OFF, ERROR, WARN, INFO, DEBUG, or ALL.

> **Note:**   Changes made at runtime using Application Server Control or JBoss AS will not be persisted. Restarting the SIP Servlet Container resets the log levels for the core components to the default values set in the global `log4j.xml` file.

For more information on these logging levels and the parameters of the SIP Servlet Container Logging MBean, see "SIP Servlet Container Logging".

Figure 10–1   Attributes of the SIP Servlet Container Logging MBean

Setting the Default Log Levels by Updating the log4j Configuration

You can also set the default log level for the core components by modifying the global `log4j.xml` file for the corresponding categories. OCMS 10.1.3.3 defines the following logger categories for its *traffic*, *system*, *config*, *badmsg* and *statstics* components:

- `oracle.sdp.ocms.traffic.`

- `oracle.sdp.ocms.system`

> **Note:** The following categories in the global `log4j.xml` file control the log level for the *system* log:
>
> ```
> <category name="oracle.sdp" additivity="false">
>   <priority value="INFO"/>
>
>   <appender-ref ref="CUSTOMER_FILE_LOG" />
>
> </category>
>
> <category name="com.hotsip" additivity="false">
>
>   <priority value="INFO"/>
>
>   <appender-ref ref="CUSTOMER_FILE_LOG" />
> ```

- `oracle.sdp.ocms.config`

- `oracle.sdp.ocms.badmsg`

- `oracle.sdp.ocms.statistics`

As illustrated in Example 10–2, you can change the default log level for the traffic component by setting the `<priority value>` for the *traffic* category (`oracle.sdp.ocms.traffic`).

Example 10–2   Changing the Default Log Level

```
<category name="oracle.sdp.ocms.traffic" additivity="false">
    <!-- change to INFO for logging of traffic -->
    <priority value="INFO"/>
    <appender-ref ref="TRAFFIC_FILE_LOG" />
  </category>
```

> **Note:** By default, the *traffic* and *badmsg* logs are set to *OFF*. As a consequence, a system using these default settings will not write any messages to the *traffic* or *badsmg* logs.

The mapping of the core components to log4j categories is located in the `sdp-log-components.xml` file. In a standalone instance of OC4J, this file is located within `ORACLE_HOME/j2ee/home/config` directory; in the JBoss Application Server, the file is located within the `JBOSS_HOME/Server/<$jboss_server_name>/conf` directory. In Oracle Application Server, this file is located at `ORACLE_HOME/J2EE/ocms/config`.

> **Note:** All instances of OC4J with a given Oracle home share the same log4j logging system configuration since the log4j logging system is deployed and configured as a global library. Applications employing log4j for logging must add the appropriate configuration to the global `log4j.xml` properties file.

Exposing a Component's Log Levels through the SIP Servlet Container Logging MBean

You can use the OCMS logging framework to configure logging for a component and then expose the log levels through the SIP Servlet Container Logging MBean. To do this, you first configure the log4j categories corresponding to the components and then map these categories to the component. For more information, see "SIP Servlet Container Logging".

Defining log4j Appenders and Categories

Before you can write logs to the log4j system from the Java classes, you must first define the log4j categories and appenders. You can configure the log4j system either programatically or by updating the global `log4j.xml` properties file (illustrated in Example 10–3).

Example 10–3   Configuring a log4j Category for a Component in log4j.xml

```
<category name="com.company.component">
   <priority value= "INFO"/>
   <appender-ref ref="CONSOLE"/>
</category>
```

Refer to `http://logging.apache.org/log4j/docs/` for more information on adding and configuring log4j categories and appenders.

Exposing the log4j Log Level for the Component

After you configure the categories in the log4j subsystem, you map them to the component by adding an entry to the `sdp-log-components.xml` file as illustrated in Example 10–4. This entry must include the name of the component and a list of the categories configured in the log4j system.

When you complete the entry, the component is exposed as an attribute of the SIP Servlet Container Logging MBean. Setting the value for this attribute sets the log level for all of the categories defined for the component in the `sdp-log-components.xml` file.

> **Note:** For these changes to take effect, the Oracle Application Server must be restarted.

Example 10–4   Mapping log4j Categories to the Component in sdp-log-components.xml

```
<component name="example component">
   <logger name= "com.company.component"/>
</component>
```

Viewing Application Log Files

You can access the J2EE-specific logging data for OMCS applications from the *Log Files* page of the Oracle 10*g* Application Server Control Console. To access this page, first click *Logs*. In the *Log Files* page that appears, expand the OCMS entry to view the deployed applications. Use the *View* task to examine the `application.log` file for an OCMS application (Figure 10–2). For more information on viewing log files, refer to *Oracle Application Server Administrator's Guide*.

Figure 10–2   Viewing Logging Data for an OCMS Application

11

Configuring Oracle Communicator

This chapter describes how administrators create customized installations of Oracle Communicator. This chapter includes the following sections:

- "Setting the Default Values for an Installation of Oracle Communicator"
- "Customizing the Installer File"
- "Setting the Oracle Communicator Upgrade Policy"
- "Configuring Oracle Communicator to Retrieve Resource-Lists Documents"
- "Enabling File Transfers"

Setting the Default Values for an Installation of Oracle Communicator

You can customize Oracle Communicator installations that populate user accounts with default settings appropriate to the configuration and topology of a particular site. For example, you can distribute installer files to Communicator users that set their accounts to use a SIP proxy with the default address of *ourproxy.example.com*. You set these defaults by creating an XML file called `customize.xml`. Using a program that supports adding files to a RAR file, you then package this file with the installer file (the self-extracting RAR file called `OracleCommunicatorSetup.exe`) that you distribute to Communicator users.

When a user installs a setup file that includes `customize.xml`, the properties from the customize file are used to overwrite the following two files in the Oracle Communicator install directory:

- Program Files\Oracle\Oracle Communicator\defaults.xml
- Program Files\Oracle\Oracle Communicator\customize\vendor.xml

`defaults.xml` is a template for creating account-specific XML files. These files are updated whenever users modify their configurations through the user interface. As a result, these properties can vary from account to account. For example, one user account-specific XML file may designate the proxy as `beta.testcomany.com`, while another may designate `proxy.example.com`.

`vendor.xml` describes the properties that are the same for all Communicator users.

If you package `customize.xml` with the installer, then the default properties defined in `vendor.xml` and `default.xml` are overwritten by those set in `customize.xml` during installation. For example, if you define an outbound proxy address in `customize.xml`, it will replace the value for the outbound proxy address in `default.xml`. All properties that are not overwritten by `customize.xml` will use default values (that is, either values already present in `defaults.xml` or `vendor.xml`, or values that are generated at runtime).

Customizing the Installer File

The Oracle Communicator installer includes a template file for `customize.xml` called `customize-sample.xml`. This file documents every property that might require customization. By modifying and renaming this file to `customize.xml` and then packaging it with the installer file as described in "Creating a Customized Installer File", you set the defaults for all of the users in an Oracle Communicator installation.

The `customize-sample.xml` file is divided in two sections: *Vendor* and *Account*. In general, *Vendor* settings apply to every account in a given Communicator installation and cannot be changed by the user. The settings configured in the *Account* section are used to create an account-specific XML file whenever a user creates a new Communicator account. Some of the properties in a user's account-specific XML file are set using Communicator's *Preferences* dialog (Figure 11–1).

Figure 11–1   The Oracle Communicator Preferences Dialog
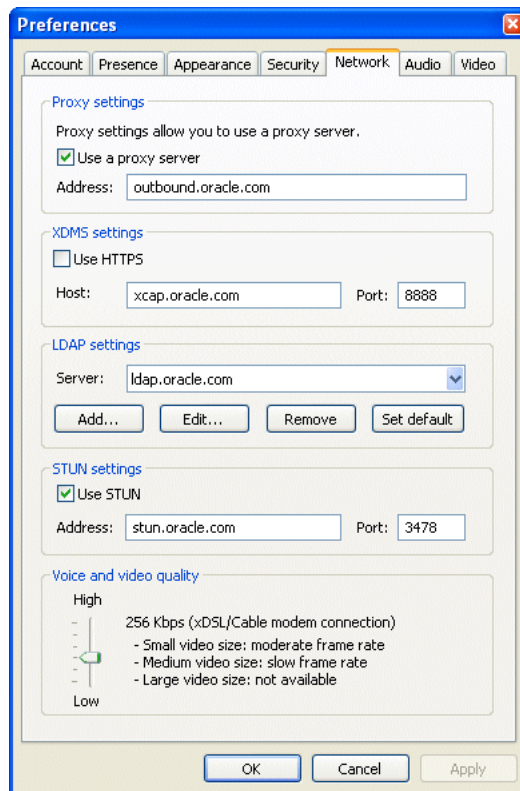
Configuring customize.xml

`customize-sample.xml` includes all of the elements needed for creating a custom installer. You can remove any element in the *Vendor* or *Account* sections that does not require customization. If you remove an XML element, the system uses a default value documented in `customize-sample.xml`.

Example 11–1 illustrates the structure of `customize-sample.xml`. For the full file, see Appendix C, "The Oracle Communicator Configuration File".

Example 11–1   customize-sample.xml

```
<Customize>
```

```
<Vendor>
   <SelfProv>...</SelfProv>
</Vendor>
<Account>
   <UseOutboundProxyAddress>...<UseOutboundProxyAddress>
   <OutboundProxyAddress>...<OutbounProxyAddress>
   <Theme>...</Theme>
   <XDMSSettings>...</XDMSSettings>
   <UseRPortForNatTraversal>...</UseRPortForNatTraversal>
   <UseStun>...</UseStun>
   <StunServerAddress>...</StunServerAddress>
   <StunServerPort>...</StunServerPort>
   <LdapServers>...</LdapServers>
   <Provisioning>...</Provisioning>
   <FileTransfer>...</FileTransfer>
   <VersionControl>...</VersionControl>
   <FileTransferEnabled>...</FileTransferEnabled>
   <UseServerResourceLists>...</UseServerResourceLists>
</Account>
</Customize>
```

Only include the elements that are appropriate to the topology and configuration of the OCMS installation.

Enabling User Self-Provisioning

The `<Vendor>` element provides Oracle Communicator users with read-only values. The properties set within this section are the same for all users. Setting the content of the `<SelfProv>` element to 1 enables end users to see the *Service Settings* menu option. This opens the service provider's Web page URL. See "Setting the Service Provider's Web Page" for more information on configuring this URL.

Setting the Outbound Proxy Address

To enable use of the outbound proxy, enter 1 as the content for `<UseOutboundProxy>`. Next, enter the IP address of the outbound proxy server where all requests are sent on the first hop. For example, enter `sip:my.host:5060;transport=tcp`. If you do not specify this value, then a default address of `outbound.<host>` is used instead, where `<host>` is taken from the Oracle Communicator user's SIP address. See also the *SIPOutboundProxy* attribute of the PresenceSupplierWebService and PresenceConsumerWebService MBeans.

Setting the Oracle Communicator Skin

The `<Theme>` element sets the default theme for the Oracle Communicator user interface. The current options are OC, Communicator, Classic, Jazz and Steel Can.

Enabling Presence

The `<XDMSSettings>` element and its child elements enable presence for Oracle Communicator users by specifying the address the XDM Server, the presence rules and hard-state settings. The child elements include:

- `<UseHttps>`

  Set to 1 to use HTTPS to connect to the Aggregation Proxy. To enable this secure connection to the server, refer to "Securing the XDMS with the Aggregation Proxy".

- `<Host>`

The host of the XDMS. For example, enter `your.xdms.domain.com`. For HTTPS, define the host for the Aggregation Proxy.

■ `<Port>`

The port number of the XDMS. For HTTPS, configure the port as the HTTPS port of the Aggregation Proxy host. For example, enter `443`.

■ `<RootContext>`

The root of the XDMS. The default value is `services`. Set `<RootContext>` to `aggregationproxy`, the context root of the Aggregation Proxy, for either HTTP or HTTPS connections to the Aggregation Proxy. If Communicator connects to the Aggregation Proxy using HTTPS (that is, `<UseHttps>1</UseHttps>`), then you must define `<RootContext>` as `aggregationproxy`. If the Aggregation Proxy is not used, then set `<RootContext>` to the default value of `services`.

■ `<PresRuleAUID>`

The ID of the application usage for presrules. The default is `pres-rules`.

■ `<PresRuleDocName>`

The name of the pres-rules document. The default value is `presrules`.

■ `<HardStateAUID>`

The ID of the application usage for PIDF (Presence Information Data Format) manipulation. The default value is `pidf-manipulation`.

■ `<HardStateDocName>`

See also "Presence" and "XCapConfig".

The document name for pidf manipulation application usage. Unauthenticated users are blocked when no rule is found. The default is `hardstate`.

■ `<ResourceListAUID>`

The application usage id (AUID) of the resource-lists document. The default value is `resource-lists`.

■ `<ResourceListsDocName>`

The name of the resource-lists document The default value is `index`.

> **Tip:** To enable Oracle Communicator to request a user's buddy lists from the XDMS, you must configure the `<ResourceListAUID>` and `<ResourceListsDocName>` elements. You must also set the `<UseServerResourceLists>` element to 1.

Enabling NAT Traversal and Discovery

The `<UseRPortForNatTraversal>`, `<UseStun>`, `<StunServerAddress>` and `<StunServerPort>` elements are used to enable NAT traversal (that is, to enable a user to connect from behind a router). The latter three properties are used to provide information on the STUN Server, if one is available. Configure these elements as follows:

■ `<UseRPortForNatTraversal>`

Set to 1 to enable use the `rport` parameter specified in RFC 3581. The `rport` parameter, which is in the *Via* header field, enables the Communicator client to request OCMS to send the response back to the source IP address and port from which the request originated.

> **Caution:** You must set `<UseRPortForNatTraversal>` to 0 for clustered configurations of Oracle Communication and Mobility Server.

- `<UseStun>`

    Set to 1 to enable STUN.

- `<StunServerAddress>`

    The address of the STUN Server.

- `<StunServerPort>`

    The primary STUN port to which to bind for listening for incoming binding requests. The value is UDP port 3478, the default STUN Port as described in RFC 3489.

For more information, see "STUN Service".

Enabling Directory Searches

The LDAP server that you define within the `<LdapServers>` element enable Oracle Communicator users to search contact lists through Oracle Communicator's support of LDAPv3. By default, Oracle Communicator does not define an LDAP Server. The child `<LdapServer>` elements have their own children, which specify the LDAP Server accessed by Oracle Communicator users. These include:

- `<Name>`

    The name of the LDAP Server.

- `<Ip>`

    The IP address of the LDAP Server.

- `<Port>`

    The port number of the LDAP Server

- `<BaseObject>`

    The starting point for searches using the DN (Distinguished Name) syntax. For example, enter `dc=oracle,dc=com` to search within oracle.com.

- `<Default>`

    Set this as the default LDAP server if more than one LDAP servers are defined.

- `<useTLS>`

    Set to 1 to use a TLS connection to the LDAP server.

- `<UseAuthentication>`

    Set to 1 to use the DN for authorization.

- `<AuthenticationAttribute>`

    The attribute to use as the DN for authorization. For example, enter `uid`.

- `<SipUriAttribute>`

    Indicates which LDAP schema attribute, if any, should be mapped to the SIP address of a contact.

- `<SipUriProtocolPrefix>`

If set to 1, this indicates that the LDAP server returns SIP addresses prepended with `sip:`.

- `<SipUriLowercaseTransform>`

  If set to 1, then the SIP address of a contact returned by the LDAP server is put in lower case.

Setting the Service Provider's Web Page

Defining the `<Provisioning>` element's child, `<Location>`, defines the service provider's Web page that Oracle Communicator user's to launch from the *Service Settings* menu item. The value for `<Location>` is a URL.

Setting the Upgrade Policy

See "Setting the Oracle Communicator Upgrade Policy".

Storing Resource-Lists Documents

To store the resource-lists document on the XMDS, set the `<UseServerResourceLists>` to 1. If the resource-lists document resides on the XDMS:

- The resource-lists document is written to the address specified by the `<XDMSSettings>` element.

- OCMS performs a full document replacement on the XDMS whenever users change their buddy lists.

- Because OCMS 10.1.3.3 does not support changes to the buddy list document through SUBSCRIBE/NOTIFY, the buddy lists for concurrent sessions of Oracle Communicator may not match. See also "Updating Buddy Lists".

To store the resource-lists document locally as a file, set `<UseServerResourceLists>` to 0.

> **Caution:** OCMS does not support switching the resource-lists document location from local file to the XDMS. If you change the location from local file to the XDMS during an upgrade, for example, then the resource-lists document is lost.

Enabling File Transfers

To enable users to send files to one-another, you must set the `<FileTransferEnabled>` element to 1 (true) to expose the **Send File** option in Oracle Communicator. Both the sender and the recipient must have this option set to 1. In addition, enter the location, port and protocol of the server that hosts the FileTransferServlet (described in "Enabling File Transfers") in the `<FileTransfer>` element's `<Location>` child element as illustrated in Example 11–2.

Example 11–2   Defining the Location of the Server That Hosts the FileTransferServlet

```
<FileTransfer>
  <Location>https://144.25.172.80:4443/filetransfer/</Location>
</FileTransfer>
```

In Example 11–2, the port is set to the secure port number 4443 to enable HTTPS.

Creating a Customized Installer File

To create the customized installer file:

1. Download and install an application that supports adding files to a RAR. These instruction use WinRAR (`http://www.rarlab.com/`) as an example.

2. Open the self-extracting RAR file, `OracleCommunicatorSetup.exe`, in WinRAR.

Figure 11–2   Opening the Oracle Communicator Setup File

3. Extract `customize-sample.xml` and rename it `customize.xml`.

4. Edit the `customize.xml` file with your specific settings. Typically, you edit all of the server settings to match the deployment. You can comment out or remove the XML elements for those properties for which the default values suffice.

5. Using WinRAR (or another application that manages RAR files), package `customize.xml` with the setup file (`OracleCommunicatorSetup.exe`) using one of the following methods:

   ■ Open `OracleCommunicatorSetup.exe` in WinRAR (Figure 11–2) and then drag and drop `customize.xml` into `OracleCommunicatorSetup.exe`. In the *Archive Name and Parameters* dialog (Figure 11–3), select `OracleCommunicatorSetup.exe` as the archive and then click **OK**.

Figure 11–3 Adding custom.xml to the Installer
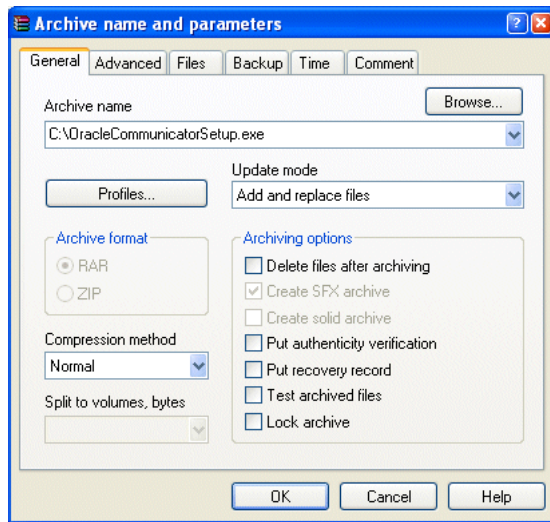
- Open `OracleCommunicatorSetup.exe` in WinRAR (Figure 11–2), click **Commands** and then select **Add Files to Archive**. In the *Select Files to Add* dialog, navigate to `customize.xml` using the **Browse** function and then select `customize.xml`. In the *Archive Name and Parameters* dialog, select `OracleCommunicatorSetup.exe` as the archive and then click **OK**. You can either add `customize.xml` back to `OracleCommunicatorSetup.exe` or rename a copy of the setup file and add `customize.xml` to that.

  > **Note:** You can also rename the `.exe` file. You can give this file any name.

6. If desired, sign the new self-extracting `.exe` so that users who download it have confidence in the source of the modified installer.

7. Distribute the modified self-extracting `.exe` to end-users.

Setting the Oracle Communicator Upgrade Policy

This section describes how to configure `customize.xml` to specify the upgrade policy for all Oracle Communicator instances installed by end users.

To set the upgrade policy, you first create an XML file, such as `upgrade.xml` (Example 11–3) that contains information regarding the site-wide recommended and required settings for Oracle Communicator. This file should reside on a Web server that is accessible to all of the installed Communicator clients. Example 11–3 illustrates the contents of this file.

Example 11–3 The Oracle Communicator Upgrade Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<Upgrade>
<Must>10.1.3.20002</Must>
<Recommend>10.1.3.20003</Recommend>
<Interval>86400</Interval>
<Download>http://example.com/myInstallInstructions.html</Download>
</Upgrade>
```

You then reference this document within `customize.xml` by defining the `<VersionControl>` element (Example 11–3) that is located in the `<Account>` section. For example, configuring the `<Location>` element to point to upgrade policy results in the retrieval and review of the upgrade policy when a user logs into Oracle Communicator.

```
<VersionControl>
   <UseVersionControl>1</UseVersionControl>
   <Location>http://policy.example.com/upgrade.xml</Location>
</VersionControl>
```

> **Note:** `upgrade.xml` and the installation instructions referenced in the `<Location>` element do not have to be located on the same server; they need only be in a locations that Communicator clients can access through HTTP. The `<Location>` element can point to a page that you create that contains downloading instructions, or it can point directly to the executable of the new installer.

If the version of the user's Communicator client is lower than the required version defined in `<Must>` element of the upgrade policy, then Communicator displays a dialog that alerts users to the fact that they must upgrade. (Figure 11–4).

Figure 11–4   The Required Upgrade Dialog
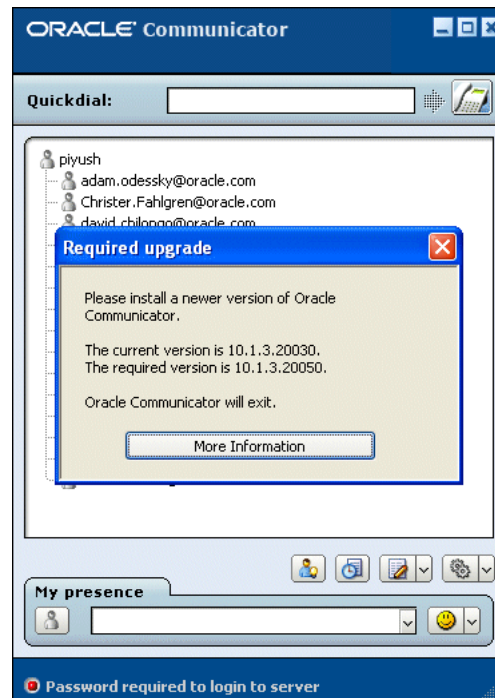
Clicking the dialog's upgrade button opens the download URL in a default URL and closes Oracle Communicator. If the user's version is higher than the required version specified in the `<Must>` element, but lower than the version defined in the `<Recommended>` element, then Communicator presents the user with a dialog with that enables the user to open the download URL from the default browser or to ignore this option.

For a Communicator installation that points to the upgrade policy described in Example 11–3, for example, Communicator does not prompt users to upgrade to versions of Communicator greater than, or equal to, both the versions set for the `<Must>` (10.1.3.20002) and `<Recommended>` (10.1.3.0003) elements and will wait the number of seconds specified in `<Interval>` (864000) before retrieving `upgrade.xml` again. The upgrade policy described in Example 11–3 would affect users running Versions 10.1.3.20001, 10.1.3.20002, 10.1.3.20003, and 10.1.3.20004 of Communicator as follows:

- 10.1.3.20001: Communicator prompts the user that an upgrade is required. When the user then clicks the button, Communicator is closed and the URL specified in `upgrade.xml` is launched in the default browser.

- 10.1.3.20002: Because the user has the latest required version, but not the latest recommended version, Communicator alerts the user that the recommended version is available (Figure 11–5). The user can then opt to close this dialog or can click a button that launches the upgrade URL in the default browser. Communicator closes when the user clicks this button.

Figure 11–5   The Recommended Upgrade Dialog
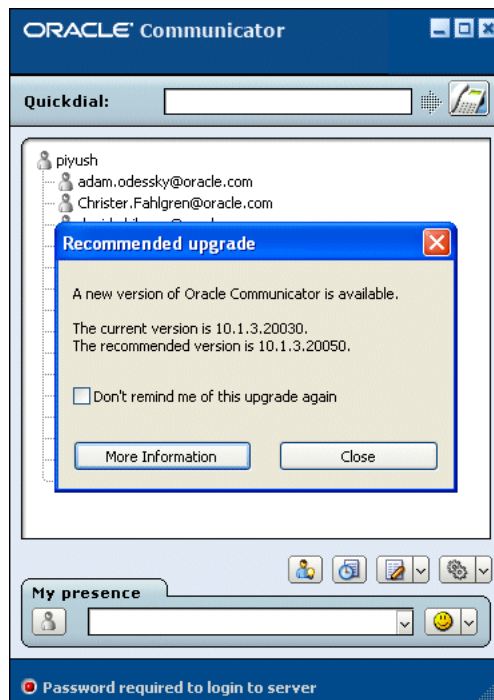
- 10.1.3.20003: Communicator does not prompt the user to upgrade.

- 10.1.3.20004: Communicator does not prompt the user to upgrade.

If `upgrade.xml` is not available for some reason (for example, the server is down, the `<VersionControl>` and `<Location>` elements were not specified properly, or a fire wall blocks access to the upgrade document), then the upgrade process will fail silently with no ill effect on the user's experience. The user can login normally.

> **Note:** It is not possible to ensure that every instance of
> Communicator running at a site matches the version specified in the
> `<Must>` element in some situations. For example, a user with a
> personal fire wall that blocks access to `uprgrade.xml` will never be
> prompted to upgrade.

Configuring Oracle Communicator to Retrieve Resource-Lists Documents

The parameters set through the `<UseServerResourceLists>`,
`<ResourceListsDocName>`, and `<ResourceListAUID>` attributes enable Oracle
Communicator to retrieve a user's buddy list from resource-lists document stored on
the XDMS at startup. Oracle Communicator reads the resource-lists document only
once, when the user first logs in. Any changes made to the resource-lists document
after that initial login (for example, through a different session of Oracle
Communicator writing to the server) are not read and may be overwritten.

Updating Buddy Lists

Oracle Communicator does not support partial updates. Instead, it saves and uploads
the entire buddy list document to the XDMS after any modification, such as:

- Adding a buddy.

- Adding a new group.

- Changing a buddy's contact information.

- Changing a group name.

- Moving a buddy to another group.

Because OCMS does not support `NOTIFY` operations (it uses `XCAP PUT` instead),
buddy lists from different instances of Oracle Communicator may not be in
synchronization. For example, Alice has two running instances of Oracle
Communicator: one on her home computer and one on her office computer. If Alice
adds Bob as a contact from her home computer, her office computer will not
automatically be notified of this change. Without restarting the office instance of
Oracle Communicator, Alice adds Charlie as a contact from her office computer.
Because the office instance of Oracle Communicator has an older XML document (one
that does not include Bob), Charlie is added to Alice's buddy list, but Bob is removed,
because OCMS replaces the entire buddy list document whenever a user modifies the
contact list in any way.

Upgrading Buddy Lists

Contacts are stored locally in Communicator 10.1.3.2; buddies are stored as VCARDS
in a `.dat` text file that is located in the user's account properties folder (typically
located at `Documents and Settings\<username>\Application`
`Data\Oracle Communicator`). Communicator 10.1.3.3 stores buddies and groups
in a new XML format which may be stored as a local file or on the XDMS. Because of
this change, Communicator 10.1.3.2 buddy lists must be upgraded to the new format.

Prompting Users to Upgrade to XDMS-Stored Buddy Lists

Once you configure the `<UseServerResourceLists>`,
`<ResourceListAUID>`, and `<ResourceListsDocName>` attributes to support

storage of the resource-lists document to the XDMS, Oracle Communicator 10.1.3.3 performs this upgrade as follows:

1. When the user logs into Oracle Communicator 10.1.3.3, Communicator determines if a `.dat` file is still present. A `.dat` file indicates the user has not yet upgraded to the 10.1.3.3 resource-lists document.

2. If a `.dat` file is present, Communicator prompts the user with a message asking if he or she wishes to upgrade their buddy list. This message is needed for users who have multiple Communicator instances, such as separate instances for home and work that store resource-lists documents on the XDMS. Because OCMS does not support partial changes or document merging, users must select a particular buddy lists to be saved. Users with multiple instances of Oracle Communicator must answer "no" to the upgrade message for every instance except for the one that the user selects to upload.

3. If the user chooses to upgrade the buddy list, any existing 10.1.3.3 buddy list is removed. The user's buddies and groups will be migrated to the resource-lists document.

   If there are no problems, the `.dat` file is given the additional `.bak` extension. If there are problems reading the `.dat` file, then Communicator informs the user that problems arose and uses an empty buddy list. Communicator renames the `.dat` file with the `.dat.bak` extension. If Communicator encounters problems when saving this file, it alerts the user that there are problems saving the file and does not rename the `.dat` file with the `.dat.bak` extension. This enables the user to the upgrade in the future. If the user chooses not to upgrade the buddy list, then the existing `.dat` file is renamed with a `.dat.bak` extension.

Note: This upgrade is only performed after users log into the new version of Oracle Communicator and are authenticated to the XDMS for the first time.

Enabling File Transfers

The `<FileTransfer>` and `<FileTransferEnabled>` elements enable Oracle Communicator users to exchange files by connecting to the FileTransferServlet. This servlet does not reside on the same OC4J container; it is instead hosted on a separate OC4J 10.1.3.3 (or 10.1.3.2) container (illustrated in Figure 11–6). See also "Deploying the FileTransferServlet"

Figure 11–6   File Transfer Topology
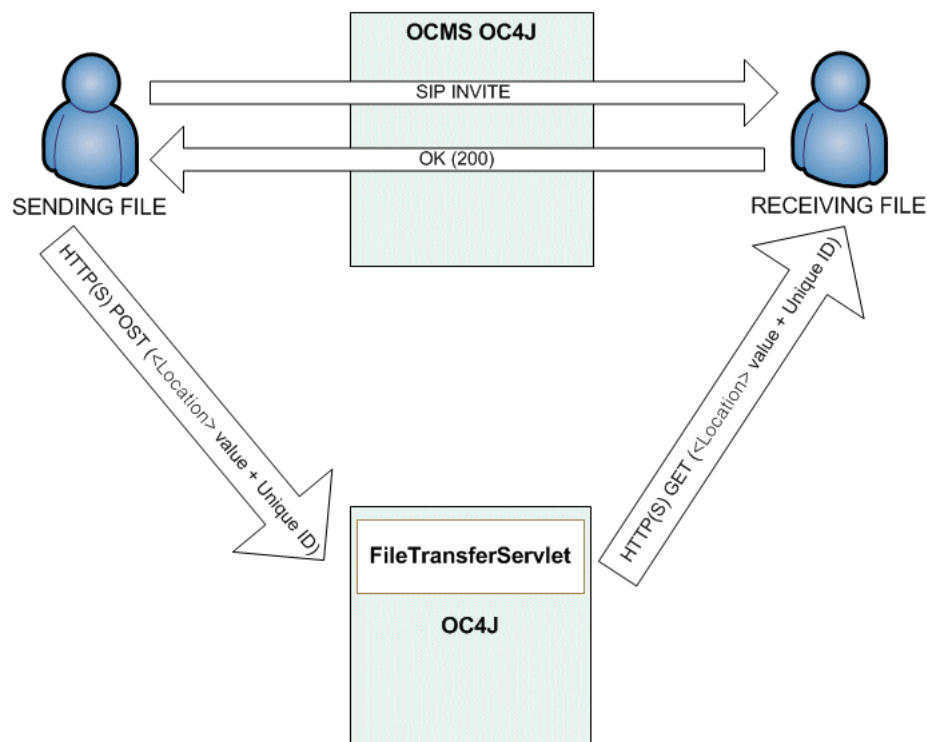

Transferring Files

Oracle Communicator enables users to send files through the following options:

- Right-clicking a contact and then selecting **Send File**.

- Entering the SIP address in the *Quick Dial* box and then selecting **Send File**.

- Selecting **File** and then **Send File** from the *Communication* dialog.

When the user selects and then sends a file, Oracle Communicator sends a SIP INVITE request to the recipient. To secure the file transfer, the INVITE contains XML that describes the file itself and the URL of the FileTransferServlet (the address designated by the `<Location>` element) and a randomly generated string.

> **Note:**   This string is unique to every file transfer; it is never re-used.

Upon receiving the INVITE request, the recipient's client alerts its user of the file transfer with the *File Offer* dialog (Figure 11–7) that indicates the name of the sender as well as the name and size of the file. If the recipient accepts the file, the client returns an OK response to the sender. The user and recipient's clients both make HTTP or HTTPS connections to the URL of the OC4J container hosting the FileTransferServlet. The servlet matches the sender to the receiver and then ushers the file to the receiver. Typically, the server hosting the FileTransferServlet does not store the files; the FileTransferServlet only conveys the files from sender to receiver.

Figure 11–7   The File Offer Dialog
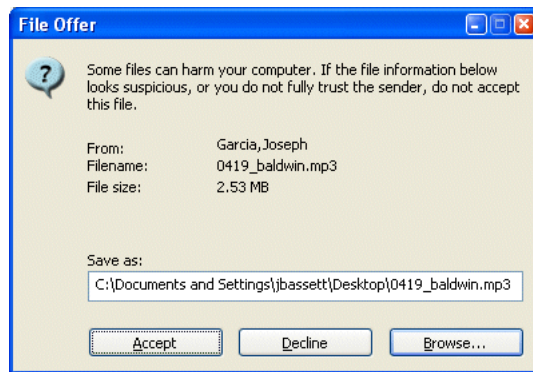
After the receiver accepts the file and selects a location to store the file, the file transfer status and any error messages display for both the sender and receiver (Figure 11–8). The recipient receives a notification once the file transfer completes and can open the selected destination folder to view the file.

If the receiver declines the file, or if the receiver's instance of Oracle Communicator is not configured to support file transfer, then a dialog notifies the sender and the file transfer ends.

Figure 11–8   File Transfer Status
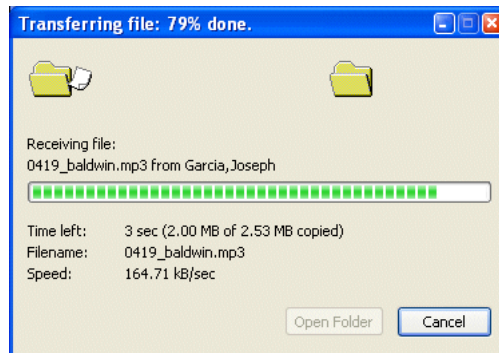
See also "Enabling File Transfers" for more information on configuring file transfers.

> **Note:**   The FileTransferServlet will timeout on a sender or receiver if a request has not been made within the period designated in the its `WEB-INF/web.xml`'s `max-connect time` parameter.

Setting the File Size

If needed, you can set a limit on the size (in bytes) of files that FileTransferServlet transfers through the `max-size` parameter of its `WEB-INF/web.xml`.

HTTP Authentication and HTTPS Certificates

Oracle Communicator 10.1.3.3 does not support authentication for file recipients since the FileTransferServlet is hosted separately from OCMS. Although Oracle Communicator 10.1.3.3 supports HTTPS, users are not prompted to accept certificates.

Deploying the FileTransferServlet

The FileTransferServlet is not installed with OCMS. It is instead deployed as an EAR (Enterprise Archive) file onto OC4J using the wizard provided in the Application Server Control Console or the `admin_client.jar` command-line tool. You can deploy the FileTransferServlet to any OC4J server that runs on Windows or Linux operating systems.

> **Note:** The OC4J server that hosts the FileTransferServlet must support both HTTP `GET` and `POST` operations.

See Oracle Communication and Mobility Server for more information on installing the FileTransferServlet. For more information on the deployment wizard, refer to "Deploying the SIP Application Using Oracle Application Server Control" and *Oracle Containers for J2EE Deployment Guide*.

A

Supported Protocols, RFCs, and Standards

This appendix lists the protocols, RFCs, and standards supported by Oracle Communication and Mobility Server in the following sections:

- "SIP Servlet Container"
- "Presence Server"

SIP Servlet Container

The following sections list the RFCs, drafts, and specification requests supported by the OCMS SIP Servlet container:

- RFCs
- Drafts
- Specification Requests

RFCs

RFC 2976 The SIP INFO Method

RFC 3261 SIP: Session Initiation Protocol

RFC 3262 Reliability of Provisional Responses in the Session Initiation Protocol

RFC 3263 Session Initiation Protocol (SIP): Locating SIP Servers

RFC 3265 Session Initiation Protocol (SIP)-Specific Event Notification

RFC 3325 Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks

RFC 3327 Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts

RFC 3428 Session Initiation Protocol (SIP) Extension for Instant Messaging

RFC 3455 Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)

RFC 3489 STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)

RFC 3581 An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing

RFC 3761 The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)

RFC 3824 Using E.164 numbers with the Session Initiation Protocol (SIP)

RFC 3840 Caller Preferences for the Session Initiation Protocol (SIP): partial support

RFC 3903 Session Initiation Protocol (SIP) Extension for Event State Publication

RFC 3966 The tel URI for Telephone Numbers

RFC 4320 Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction

RFC 4321 Problems Identified Associated with the Session Initiation Protocol's (SIP) Non-INVITE Transaction

Drafts

campen-sipping-stack-loop-detect An Efficient Loop Detection Algorithm for SIP Proxies

lawrence-maxforward-problems Problems with Max-Forwards Processing (and Potential Solutions)

ietf-sip-connect-reuse Connection Reuse in the Session Initiation Protocol (SIP)

sparks-sipping-max-breadth Limiting the Damage from Amplification Attacks in SIP Proxies

ietf-sip-fork-loop-fix Addressing an Amplification Vulnerability in Forking Proxies

Specification Requests

JSR 116 SIP Servlet 1.0

Presence Server

The following sections list protocols, RFCs, drafts, and standards supported by the OCMS Presence Server and its components:

- RFCs
- Drafts Referenced in the Composition Policies
- XDMS Server
- Authorization and Privacy Filtering
- Presence Data Modeling and Processing
- OMA Extensions
- Hard State via XCAP
- Watcher Filtering

RFCs

RFC 2778 A Model for Presence and Instant Messaging

RFC 2779 Instant Messaging / Presence Protocol Requirements

RFC 3265 Session Initiation Protocol (SIP)-Specific Event Notification

RFC 3856 A Presence Event Package for the Session Initiation Protocol (SIP)

RFC 3857 A Watcher Information Event Template-Package for the Session Initiation Protocol (SIP)

RFC 3858 An Extensible Markup Language (XML) Based Format for Watcher Information

RFC 3859 Common Profile for Presence (CPP)

RFC 3863 Presence Information Data Format (PIDF)

RFC 3903 Session Initiation Protocol (SIP) Extension for Event State Publication

RFC 4119 A Presence-based GEOPRIV Location Object Format

Drafts Referenced in the Composition Policies

The following drafts are transparent to the server, but are important for clients. These drafts are specifically referenced in composition policies such as the OMA composition policy.

"RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)", draft-ietf-simple-rpid-10

"CIPID: Contact Information in Presence Information Data Format", draft-ietf-simple-cipid-07

"Timed Presence Extensions to the Presence Information Data Format (PIDF) to Indicate Presence Information for Past and Future Time Intervals", draft-ietf-simple-future-05

Partial Publish and Partial Notify

draft-ietf-simple-partial-notify

draft-ietf-simple-partial-pidf-format

draft-ietf-simple-partial-publish

XDMS Server

"The Extensible Markup Language (XML) Configuration Access Protocol (XCAP)", draft-ietf-simple-xcap-08

"A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-07

"A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages", draft-ietf-sip-content-indirect-mech-05

Exceptions:

The current XCAP application does not support partial document put or get.

Authorization and Privacy Filtering

"A Document Format for Expressing Privacy Preferences", draft-ietf-geopriv-common-policy-05

"Presence Authorization Rules", draft-ietf-simple-presence-rules-04

draft-ietf-geopriv-common-policy-05

Presence Data Modeling and Processing

"A Data Model for Presence", draft-ietf-simple-presence-data-model-07

"A Processing Model for Presence", draft-rosenberg-simple-presence-processing-model-01

OMA Extensions

[urn:oma:params:xml:ns:pidf:oma-pres] OMA extensions to the presence data model (OMA-TS-Presence_SIMPLE-V1_0-20051122-C)

[urn:oma:params:xml:ns:common-policy] OMA extensions to geopriv common policy (OMA-TS-XDM_Core-V1_0-20051122-C)

[urn:oma:params:xml:ns:pres-rule] OMA extensions to presence rules (OMA-TS-Presence_SIMPLE_XDM-V1_0-20051122-C)

Hard State via XCAP

"An Extensible Markup Language (XML) Configuration Access Protocol (XCAP) Usage for Manipulating Presence Document Contents", draft-ietf-simple-xcap-pidf-manipulation-usage-02

Watcher Filtering

draft-ietf-simple-event-filter-funct

draft-ietf-simple-filter-format

Diameter

Following are the standards and specifications supported by Diameter:

ID-NAS Diameter Network Access Server Application Internet Draft draft-ietf-aaa-diameter-nasreq-17.txt

RCF3589 Diameter command code for third generation partnership project (3GPP) release 5

RFC2975 Introduction to accounting management

RFC3539 Authentication, Authorization and Accounting (AAA) Transport Profile

RFC3588 Diameter Base Protocol

RFC4006 Diameter Credit Control Application

TS23.002 Network architecture

TS23.228 IP Multimedia Subsystem (IMS); Stage 2

TS29.230 Diameter applications; 3GPP specific codes and identifiers

TS29.328 IP Multimedia (IM) Subsystem Sh interface; Signaling flows and message contents

TS29.329 Sh Interface based on the Diameter protocol; Protocol details

TS32.200 Telecommunication management; Charging management; Charging principles

TS32.225 Telecommunication Management; Charging Management; Charging data description for the IP Multimedia Subsystem (IMS)

TS32.240 Telecommunication management; Charging management; Charging architecture and principles

TS32.260 Telecommunication management; Charging Management; IP Multimedia Subsystem (IMS) charging

TS32.299 Telecommunication Management; Charging Management; Diameter charging applications

B

Configuring Oracle Internet Directory as the User Repository

This appendix describes how to configure Oracle Internet Directory (OID), the LDAP data store used by Oracle WebCenter Suite, as the user provisioning repository for an OCMS deployment. This appendix includes the following sections:

- "Overview of Configuration for OID Support"
- "Configuring the OID LDAP Backend"
- "Repackaging Subscriber Data Services"
- "Provisioning OCMS Users to OID"

Overview of Configuration for OID Support

For OCMS to support authentication and authorization services for users provisioned to OID requires the following configuration for both OCMS and OID:

- "Configuring the OID LDAP Backend"
- "Repackaging Subscriber Data Services"

Prerequisites for OID Support

Using the OID data store requires the following:

- A properly installed and configured instance of OCMS.
- An instance of OID, Version 10.1.4.0.1.
- You must enable reversible password encryption for the LDAP realms employed for user authentication by selecting *Userpassword Reversible Encryption*. For more information, see *Oracle Internet Directory Administrator's Guide*.

> **Note:** OID is only supported for standalone OCMS deployments and OCMS deployments on Version 10.1.3.2 or higher of Oracle Application Server. OCMS does not support OID on JBoss Application Server deployments. See *Oracle Communication and Mobility Server Installation Guide* for further hardware and software requirements and installation options.

Configuring the OID LDAP Backend

In addition to configuring reversible password encryption (described in "Prerequisites for OID Support"), you must also configure the following OID LDAP attributes for the OID LDAP backend:

- *orclcommonnicknameattribute* (See "Mapping JAAS Usernames to LDAP User Entries".)

- *orclsubscribernicknameattribute* (See "Mapping JAAS Realms to LDAP Subscribers".)

- *orclcommonnamingattribute* (See "Mapping JAAS Roles to LDAP Groups".)

Mapping 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, cd=Products, cn=OracleContext* for each of the provisioned LDAP realms. For example, setting this attribute to *uid* for a given realm implies that SIP or Web users authenticating against OID must provide their corresponding LDAP UID as their username during authentication.

Mapping 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* for an OID deployment implies that SIP or Web users authenticating against OID must belong to the JAAS realm identified by the value of the *o* attribute. As a result, user *sip.user@company.com* is challenged under the realm, *company*. The mapping of SIP domains to JAAS realms is exposed through the SipServletContainer's *DomainsAndRealms* attribute. In this example, the SIP domain, *company.com*, is mapped to the JAAS realm, *company*. The JAAS realm, *company*, is then mapped to the LDAP Subscriber for whom the value for the attribute in *orclsubscribernicknameattribute* (that is, the *o* attribute) is set to *company*. See also "SIP Servlet Container".

Mapping 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=us, dc=com* and the *orclcommonnamingattribute* is set to *cn*, then that JAAS user is populated with the "Location Service" JAAS role.

Repackaging Subscriber Data Services

Configuring OCMS to support OID requires that Subscriber Data Services (`subscriberdataservices.ear`) and its child applications be undeployed from the OCMS OC4J instance. Before the application and its child applications can be re-deployed, the user service and security service EJB configuration must be altered by adding the following LDAP configuration parameters to the `ejb-jar.xml` files for `securityservice.jar` and `userservice.jar`:

- `java.naming.security.principal`

- `java.naming.provider.url`

- `java.naming.security.protocol` (an optional parameter)

The user service EJB configuration also exposes the `SipUriLdapAttribute`, which defines the LDAP user attribute where the SIP URI is stored. This attribute defaults to *mail* if no value is defined.

Configuring User Service and Security Service

To configure the Subscriber Data Services application with OID as the user provisioning store:

1. Copy the EAR file of the Subscriber Data Services application (`subscriberdataservices.ear`) as well as its child applications to a temporary directory.

2. Undeploy the Subscriber Data Services application and its child applications from the OC4J instance. See "Deploying, Undeploying, and Redeploying SIP Servlet Applications with Application Server Control".

3. Expand the Subscriber Data Services application in the temporary directory.

4. Expand `securityservice.jar`

5. Edit `ejb-jar.xml` (located under META-INF) by replacing the following entry with the entry listed in Example B–1 that includes the `java.naming.security.principal`, `java.naming.provider.url`, and the `java.naming.security.protocol` parameters.

```
<env-entry>
   <description><![CDATA[Datasource for Service
activationfacades]]></description>
   <env-entry-name>SecurityServiceDSN</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[java:jdbc/OcmsSsDs]]></env-entry-value>
   </env-entry>
   <env-entry>
       <env-entry-name>SecurityDAOImpl</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>

<env-entry-value><![CDATA[com.hotsip.securityservice.dao.timesten.SecurityDAOIm
pl]]>
 </env-entry-value>
 </env-entry>
```

Example B–1   ejb-jar.xml Entries

```
<env-entry>
   <description><![CDATA[LDAP Admin User]]></description>
   <env-entry-name>java.naming.security.principal</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>

<env-entry-value><![CDATA[cn=orcladmin,cn=Users,dc=us,dc=oracle,dc=com]]></env-ent
ry-value>
</env-entry>
   <env-entry>
   <description><![CDATA[LDAP Admin User Credentials]]></description>
   <env-entry-name>java.naming.security.credentials</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[!welcome1]]></env-entry-value>
</env-entry>
<env-entry>
```

```
    <description><![CDATA[LDAP Provider URL]]></description>
    <env-entry-name>java.naming.provider.url</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>

<env-entry-value><![CDATA[ldap://ldapusers.company.com:636]]></env-entry-value>
</env-entry>
    <env-entry>
    <description><![CDATA[Security Protocol (e.g. ssl)]]></description>
    <env-entry-name>java.naming.security.protocol</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value><![CDATA[ssl]]></env-entry-value>
</env-entry>
<env-entry>
   <env-entry-name>SecurityDAOImpl</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>

<env-entry-value><![CDATA[com.hotsip.securityservice.dao.ldap.SecurityDAOImpl]]>
</env-entry-value>
</env-entry>
```

> **Note:** The `java.naming.security.principal`,
> `java.naming.security.credentials` and
> `java.naming.provider.url` environment entries must be
> updated with the LDAP server's configuration. In addition, the
> optional entry, `java.naming.security.protocol`, must be set to
> `"ssl"` for SSL-based connections to the OID LDAP server.

6. Repackage `securityservice.jar`.

7. Expand the `userservice.jar`.

8. Edit the `ejb-jar.xml` under META-INF by replacing the following entry with
   the entry described in Example B–2 that includes the
   `java.naming.security.principal`, `java.naming.provider.url`,
   `java.naming.security.protocol`, and `SipUriLdapAttribute`
   parameters.

```
<env-entry>
    <description><![CDATA[Datasource for Service activation
facades]]></description>
    <env-entry-name>UserServiceDSN</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value><![CDATA[java:jdbc/OcmsUsDs]]></env-entry-value>
</env-entry>
     <env-entry>
     <env-entry-name>UserDAOImpl</env-entry-name>
     <env-entry-type>java.lang.String</env-entry-type>
<env-entry-value><![CDATA[oracle.sdp.userservice.dao.timesten.UserDAOImpl]]>
</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>StoreHashedCredentials</env-entry-name>
    <env-entry-type>java.lang.Boolean</env-entry-type>
    <env-entry-value><![CDATA[True]]></env-entry-value>
</env-entry>
```

Example B–2   userservice.jar Entries

```
<env-entry>
   <description><![CDATA[LDAP Admin User]]></description>
   <env-entry-name>java.naming.security.principal</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>

<env-entry-value><![CDATA[cn=orcladmin,cn=Users,dc=us,dc=oracle,dc=com]]></env-ent
ry-value>
</env-entry>
<env-entry>
   <description><![CDATA[LDAP Admin User Credentials]]></description>
   <env-entry-name>java.naming.security.credentials</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[welcome1]]></env-entry-value>
</env-entry>
<env-entry>
   <description><![CDATA[LDAP Provider URL]]></description>
   <env-entry-name>java.naming.provider.url</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[ldap://ldapusers.company.com:636]]></env-entry-value>
</env-entry>
<env-entry>
   <description><![CDATA[The LDAP user attribute containing the user's SIP
URI.]]></description>
   <env-entry-name>SipUriLdapAttribute</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[mail]]></env-entry-value>
</env-entry>
    <env-entry>
    <description><![CDATA[Security Protocol (e.g. ssl)]]></description>
    <env-entry-name>java.naming.security.protocol</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value><![CDATA[ssl]]></env-entry-value>
</env-entry>
<env-entry>
   <env-entry-name>UserDAOImpl</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value><![CDATA[oracle.sdp.userservice.dao.ldap.UserDAOImpl]]>
   </env-entry-value>
</env-entry>
```

> **Note:** The `java.naming.security.principal`,
> `java.naming.security.credentials` and
> `java.naming.provider.url` environment entries must be
> updated with the LDAP server's configuration. In addition, the
> optional entry, `java.naming.security.protocol`, must be set to
> `"ssl"` for SSL-based connections to the OID LDAP server. The
> userservice exposes an optional environment parameter,
> `SipUriLdapAttribute`. The value set for this entry is the LDAP
> user attribute where the SIP URI is stored. If no value is set for this
> entry, then the attribute defaults to `mail`.

9. Repackage `userservice.jar`.

10. Repackage the EAR file for Subscriber Data Services.

11. Redeploy the repackaged Subscriber Data Services application and its child applications to the OCMS OC4J instance.

Provisioning OCMS Users to OID

You cannot provision users to OID using Sash. Instead, you must provision user accounts to OID using the Oracle Identity Management (OIM) Web-based Oracle Delegated Administration Services application (OIDDAS), described in *Oracle Identity Management Guide to Delegated Administration*. You access this tool from a browser by entering `http://<host>:<port>/oiddas`, where the host and port are the hostname and HTTP port for the Oracle Application Server instance of the OID deployment.

Adding Users to LDAP Groups

Because certain OCMS applications require specific JAAS roles, users must belong to the correct LDAP groups. For example, users of the Proxy Registrar application must belong to the "Location Services" group. You can add users to LDAP groups either through the Oracle Delegated Administration Services application or the oidadmin LDAP utility. Using Oracle Delegated Administration Services, you can add a user to an LDAP group by adding the distinguished name of the user to the list of values in the LDAP group's *uninquemember* attribute. For more information, refer to *Oracle Identity Management Guide to Delegated Administration.*

C

The Oracle Communicator Configuration File

This appendix includes the `customize-sample.xml` file for Oracle Communicator described in Chapter 11, "Configuring Oracle Communicator".

The customize-sample.xml File

`customize-sample.xml` (Example C–1) is the template file for `customize.xml`.

Example C–1   The customize-sample.xml File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Customize>
- <!--
 in order to provide default settings for the installation,
     this file should be edited and renamed customize.xml,
     then included in the Oracle Communicator installation
     self-extracting RAR (i.e. OracleCommunicatorSetup.exe)

     for true/false values, 0 means false, 1 means true

     it is expected that most administrators will need
     to modify at least the proxy and XDMS settings to
     reflect the site topology, and that most administrators
     will choose to modify the STUN and LDAP settings
     as well

     this file lists default values; any property in this
     file may be removed with no impact on the installation;
     it is recommended that an administrator only keep
     those properties for which they intend to use
     non-default values


  -->
- <!--
 the Vendor section contains properties that will
       remain the same for all user accounts in the
       installation and is not editable through the
       UI by the user

  -->
- <Vendor>
- <!--
 if 1, the "Service Settings" menu option will
       be visible, which will open the URL specified
       in Account/Provisioning/Location in a browser
```

```
  -->
  <SelfProv>0</SelfProv>
  </Vendor>
- <Account>
- <!--  whether or not to use the outbound proxy
  -->
  <UseOutboundProxy>1</UseOutboundProxy>
- <!--
 the address of the proxy server, preceded by sip:
     if this is not specified, the default is
     outbound.<host> where <host> comes from the user's
     SIP address

  -->
  <OutboundProxyAddress>sip:outbound.example.com</OutboundProxyAddress>
- <!--  the default UI theme to use
  -->
  <Theme>OC</Theme>
- <!--
 the XDMS server, presence rule and hard state settings;
     if this is not specified the default for the Host is
     xcap.<host> where <host> comes from the user's SIP address;
     the other values are the same as default except RootContext
     defaults to services instead

  -->
- <XDMSSettings>
  <UseHttps>0</UseHttps>
  <Host>xcap.example.com</Host>
  <Port>8888</Port>
  <RootContext>aggregationproxy</RootContext>
  <PresRuleAUID>pres-rules</PresRuleAUID>
  <PresRuleDocName>presrules</PresRuleDocName>
  <HardStateAUID>pidf-manipulation</HardStateAUID>
  <HardStateDocName>hardstate</HardStateDocName>
  <ResourceListsAUID>resource-lists</ResourceListsAUID>
  <ResourceListsDocName>index</ResourceListsDocName>
  </XDMSSettings>
- <!--
 whether to use rport as described in RFC 3581 for
     NAT traversal; the default is 0; rport may
     not be supported by OCMS in a cluster environment,
     in which case this should be 0

  -->
  <UseRPortForNatTraversal>1</UseRPortForNatTraversal>
- <!--  the STUN server
  -->
  <UseStun>0</UseStun>
  <StunServerAddress>stun.example.com</StunServerAddress>
  <StunServerPort>3478</StunServerPort>
- <!--
 the LDAP server(s) to use for contact searching; by
     default no LDAP server is provided

  -->
- <LdapServers>
- <LdapServer>
  <Name>ldap.example.com</Name>
```

```
  <Ip>ldap.example.com</Ip>
  <Port>389</Port>
  <BaseObject>dc=example,dc=com</BaseObject>
  <Default>yes</Default>
  <UseTLS>0</UseTLS>
  <UseAuthentication>0</UseAuthentication>
- <!--
 the attribute to use as the distinguished name
     for authorization

 -->
  <AuthorizationAttribute>uid</AuthorizationAttribute>
- <!--
 the SipUriAttribute indicates which
     LDAP schema attribute, if any, should be
     mapped to the SIP address of a contact

 -->
  <SipUriAttribute>mail</SipUriAttribute>
- <!--
 if true, indicates the SIP address
     of the contact returned by the LDAP server
     will have the sip: protocol prepended

 -->
  <SipUriProtocolPrefix>0</SipUriProtocolPrefix>
- <!--
 if true, this will cause the SIP address
     of the contact returned by the LDAP server
     to be made lowercase

 -->
  <SipUriLowercaseTransform>1</SipUriLowercaseTransform>
  </LdapServer>
  </LdapServers>
- <!--
 the URL to launch if the user selects "Service Settings"; that
     option will only be available to the user if Vendor/SelfProv
     is 1

 -->
- <Provisioning>
  <Location>http://example.com/selfprovisioning</Location>
  </Provisioning>
- <!--
 enables/disables VersionControl and provides the URL
             to the upgrade policy XML document; see the
             Administrator's guide for more information;
             by default, VersionControl is disabled

 -->
- <VersionControl>
  <UseVersionControl>1</UseVersionControl>
  <Location>http://example.com/communicator/upgrade.xml</Location>
  </VersionControl>
- <!--
 enables/disables the file transfer functionality; in order
     to support file transfer you must have the FileTransferServlet
     deployed; the Location should be the full URL of the
     FileTransferServlet (e.g. http://example.com/filetransfer)
```

```
-->
<FileTransferEnabled>0</FileTransferEnabled>
- <FileTransfer>
<Location />
</FileTransfer>
</Account>
</Customize>
```

D

Third-Party Licensing

This Appendix lists third-party licensing information for software included in this release.

Third-Party Licenses

Third-party software license information for software included in this release is listed in Table D–1.

Table D–1   Third-party licenses

Library/Component	License
Saxon B	http://www.mozilla.org/MPL/MPL-1.1.html
Commons HTTPClient 2.0, 3.0	http://apache.org/licenses/
Commons Logging1.0.3, 1.1	http://apache.org/licenses/
Commons Codec 1.3	http://apache.org/licenses/
Commons Lang 2.1	http://apache.org/licenses/
Log4J 1.2.9	http://apache.org/licenses/
XMLBeans (xbean.jar) 2.1.0	http://apache.org/licenses/
Commons collection 3.1	http://apache.org/licenses/
DNSJava 2.0.1	View DNSJava License
Jline 0.9.1	View Jline License
Jdom 1.0	View Jdom License
Velocity1.4	http://apache.org/licenses/
JSR116	View JSR116 License
aopalliance-1.0.jar 1.0	http://apache.org/licenses/
spring 1.2.7	http://apache.org/licenses/
stax-api (JSR 173) 1.0	http://apache.org/licenses/
jainsip (JSR 32) 1.2	http://www.jcp.org/en/home/index
Jaxen	http://jaxen.org/license.html
Saxpath	http://jaxen.org/license.html
Boost	http://www.boost.org/LICENSE_1_0.txt

Table D–1 (Cont.) Third-party licenses

Library/Component	License
Wtl	http://www.microsoft.com/resources/sharedsource/licensingbasics/permissivelicense.mspx
RSA	/* Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.
	License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest
	Algorithm" in all material mentioning or referencing this software or this function.
	License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.
	RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.
	These notices must be retained in any copies of any part of this documentation and/or software. */
RFC 2617	http://rfc.net/rfc2617.html
	IETF's IPR policy is at
	http://www.ietf.org/rfc/rfc3978.txt and
	http://www.ietf.org/rfc/rfc3978.txt
GUI header file (atlgdix.h, menubutton.h)	Additional GDI/USER wrappers
	Written by Bjarke Viksoe (bjarke@viksoe.dk)
	Copyright (c) 2001-2002 Bjarke Viksoe.
	Thanks to Daniel Bowen for COffscreenDrawRect.
	This code may be used in compiled form in any way you desire. This file may be redistributed by any means PROVIDING it is not sold for profit without the authors written consent, and providing that this notice and the authors name is included.
	This file is provided "as is" with no expressed or implied warranty.
	The author accepts no liability if it causes any damage to you or your computer whatsoever. It's free, so don't hassle me about it.
	Beware of bugs.
String handling (stdstring.h)	COPYRIGHT:
	1999 Joseph M. O'Leary. This code is free. Use it anywhere you want.
	Rewrite it, restructure it, whatever. Please don't blame me if it makes your $30 billion dollar satellite explode in orbit. If you redistribute it in any form, I'd appreciate it if you would leave this notice here.
	If you find any bugs, please let me know:
	jmoleary@earthlink.net
	http://home.earthlink.net/~jmoleary
GUI Slider implementation (slider.js)	Slider000118 by Christiaan Hofman, January 2000
	You may use or modify this code provided that this copyright notice appears on all copies.

Index

Subscriber Data Services, 1-20, 9-23
system usage statistics, 9-12

T

testing environment, 3-10
third-party load balancer, 3-2
transport protocol, 4-3

U

Upgrading SIP applications, 8-12
User Data, 1-21
users
    authentication against RADIUS authentication
        system, 6-2
    authentication against TimesTen In-Memory
        database, 6-2
    bulk provisioning, 5-14
    provisioning to the TimesTen In-Memory database
        using the CommandService Mbean, 5-9
    provisioning users to the TimesTen In-Memory
        database using Sash, 5-7

W

Watcher, 1-16
web.xml, 8-9, 8-12

X

XDMS
    provisioning with Sash, 5-11
    provisioning with the CommandService
        MBean, 5-11
XDMS Server, 3-8