Oracle® Communications USM
Configuration Guide

Release S-CZ7.3.5

May 2018

ORACLE

Oracle Communications USM Configuration Guide, Release S-CZ7.3.5

Contents

3 System Configuration

5 Oracle USM Supporting the IMS Core

6 Routing with Local Policy

8 Local Subscriber Tables

9 Third Party Registration

10 Oracle USM Supporting the IMS Edge

11 SIP Signaling Services

12 Number Translation

13 Admission Control and QoS

14 Static Flows

15 High Availability Nodes

16 Security

17 Lawful Intercept

18 External Policy Servers

19 Transcoding

21 Tunneled Services Control Function

22 RCS Services

23 References and Debugging

A RTC Support

B USM Base Configuration Elements

About This Guide

This ACLI Configuration Guide provides information about:

- Basic concepts that apply to the key features and abilities of your Oracle USM
- Information about how to load the Oracle USM system software image you want to use and establish basic operating parameters
- System-level functionality for the Oracle USM
- Configuration of all components of the Oracle USM

Supported Platforms

Release Version S-CZ7.3.5 includes both the Oracle Core Session Manager (CSM) and Unified Session Manager (USM) products. The Oracle USM is supported on the Acme Packet 4500, 4600, 6100, and 6300 series platforms. The Oracle CSM is supplied as virtual machine software or as a software-only delivery suitable for operation on server hardware. Refer to sales documentation updates for information further specifying hardware support.

Related Documentation

The following table lists the members that comprise the documentation set for this release:

Document Name	Document Description
Acme Packet 4500 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4500.
Acme Packet 4600 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 4600.
Acme Packet 6100 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 6100.
Acme Packet 6300 Hardware Installation Guide	Contains information about the components and installation of the Acme Packet 6300.
Release Notes	Contains information about the current documentation set release, including new features and management changes.
ACLI Configuration Guide	Contains information about the administration and software configuration of the Service Provider Oracle USM.
ACLI Reference Guide	Contains explanations of how to use the ACLI, as an alphabetical listings and descriptions of all ACLI commands and configuration parameters.
Maintenance and Troubleshooting Guide	Contains information about Oracle USM logs, performance announcements, system management, inventory management, upgrades, working with configurations, and managing backups and archives.

Document Name	Document Description
MIB Reference Guide	Contains information about Management Information Base (MIBs), Oracle Communication's enterprise MIBs, general trap information, including specific details about standard traps and enterprise traps, Simple Network Management Protocol (SNMP) GET query information (including standard and enterprise SNMP GET query names, object identifier names and numbers, and descriptions), examples of scalar and table objects.
Accounting Guide	Contains information about the Oracle USM's accounting support, including details about RADIUS and Diameter accounting.
HDR Resource Guide	Contains information about the Oracle USM's Historical Data Recording (HDR) feature. This guide includes HDR configuration and system-wide statistical information.
Administrative Security Essentials	Contains information about the Oracle USM's support for its Administrative Security license.
Security Guide	Contains information about security considerations and best practices from a network and application security perspective for the Oracle USM family of products.
Installation and Platform Preparation Guide	Contains information about upgrading system images and any pre-boot system provisioning.
Call Traffic Monitoring Guide	Contains information about traffic monitoring and packet traces as collected on the system. This guide also includes WebGUI configuration used for the SIP Monitor and Trace application.

Hardware documentation is relevant only to the Oracle USM. Refer to your hardware vendor's documentation for information required for Oracle CSM operation.
Version SCZ735 software relies on version SCZ730 documentation for some documentation. This documentation includes:

- The ACLI Reference Guide
- The Troubleshooting and Maintenance Guide
- The Administrative Security Essentials Guide

Revision History

Date	Description
March 2016	• Initial Release
June 2016	• Added a note in the section Tunnel mode vs Transport mode under IPSec support • Edited section IPSec support • Clarifies Static Flows as unidirectional • Corrects explanation of diameter manipulation
September 2016	• Adds new scenario under Fax Transcoding topic
December 2016	• Defect fix ACMECSBC-22254
January 2017	• Adds a note to "Configurable Alarm Thresholds and Traps" regarding a change to critical memory alarm threshold functionality. • Updates Ingress Policy in Transcoding

Date	Description
February 2017	• Corrects show sipd agents <agent name> output example • Adds port allocation details to Pooled Transcoding
March 2017	• Adds a new section called "TACAS+ Faults". • Provides a more thorough description of the **refer-to-uri-prefix** option in *Global SIP Options*. • Adds notes to clarify the **media-policy > tos-settings** parameter is not RTC supported.
May 2017	• Updates "Session Agents Session Groups and Local Policy" for clarity. • Corrects various typographical errors.
September 2017	• Corrects various typographical errors.
October 2017	• Renames "Port 3000 and 3001 Configuration" to "Enable System to Connect to SDM.
November 2017	• Updates the "Minimum Advertised SSL/TLS Version" and "Configuring a TLS Profile" sections for accuracy. • Updates "REFER-Initiated Call Transfer" to reflect that the **dyn-refer-term** parameter is available under **realm-config** only. • Removes references to the deprecated MIKEY feature.
January 2018	• Updates the "Security" chapter's "Process Overview" section for accuracy.
May 2018	• Corrects the default and valid range of values for the "process-log-port" parameter in "Configure the Process Log Server."

1
Oracle USM Basics

This chapter introduces key features and capabilities of the Oracle USM. As an integrated IP Multimedia Sub-system (IMS) Call Session Control Function (CSCF), the Oracle USM performs CSCF roles on a single platform. This configuration can simplify IMS deployment and operation scenarios. See product specification or Oracle consultative services to understand applicable deployments. This document does not provide general instruction on IMS; review the many generally-available IMS resources if you need introduction to its complex mechanisms.

The Oracle USM operates within the context of an IMS or over-the-top (OTT) deployment. Within IMS, the device performs standard registration and call handling functions in compliance with 3GPP requirements. The presence of multiple CSCF functions on a single platform can simplify IMS operations. For those who prefer an OTT deployment, the Oracle USM supports registration and call management via an ENUM or local subscriber database.

The Oracle USM functions within IMS environments using Home Subscriber Server (HSS) deployments as subscriber database. Operation with IMS HSS resources is described in the Diameter Based Oracle USM chapter. For OTT, operation using ENUM resources is described in the ENUM Based Oracle USM chapter and operation with local subscriber tables is described in the Local Subscriber Tables chapter.

By utilizing Oracle's Session Border Controller (SBC) product features, the Oracle USM is also capable of performing a wide range of edge functions, separated along the following roles:

- Access functions
- Interconnect functions

This functionality and configuration is covered in the Oracle USM Supporting the IMS Edge chapter.

Oracle USM and IMS

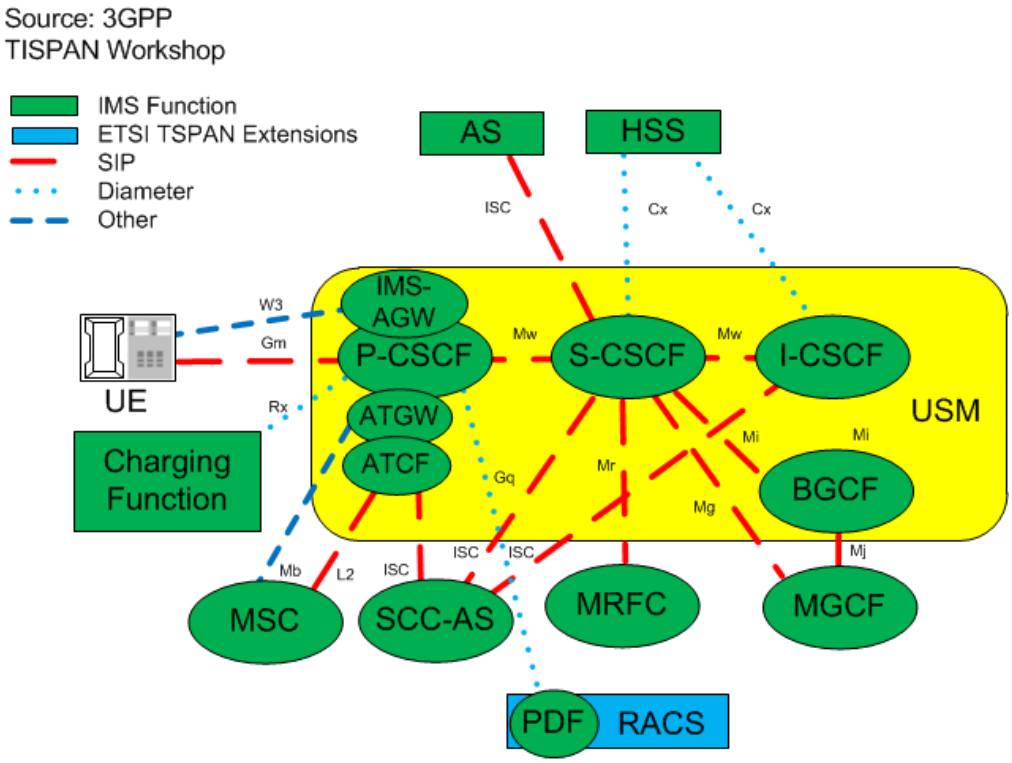
The ETSI TISPAN NGN defines several subsystems that make up the NGN architecture. The model for the target NGN architecture is depicted below.

The Oracle USM is designed to function as an integrated:

- Proxy-Call Session Control Function (P-CSCF)
- Interrogating-Call Session Control Function (I-CSCF)
- Serving-Call Session Control Function (S-CSCF)

In addition, the Oracle USM performs many related functions. As such, the Oracle USM is typically deployed in support of LTE and other IMS-based networks.

The functions performed by the Oracle USM are best understood as functions of standard IMS elements. The diagram below depicts the interconnection of these elements across an IMS architecture.



High level definitions of these functions include:

- P-CSCF—The first point of contact within the IMS system for the endpoint. The P-CSCF determines whether and how to pass the traffic to the appropriate S-CSCF.
- I-CSCF—IMS passes traffic to the I-CSCF if the target S-CSCF is unknown.
- S-CSCF—Interaction with the Home Subscriber Server (HSS) determines whether and how to provide service to the endpoint.
- BGCF—The breakout gateway control function provides signaling transit to network domains external to the IMS.
- IMS-AGW—Access gateway complimenting the P-CSCF function to broaden the range of devices that can access the IMS.
- Ancilliary Access Functions—As part of 3GPP Release 10, new interfaces and protocols have been defined to improve mobility across LTE and 2G/3G networks and address the latency concern from previous architectures. The enhancements to the SRVCC system defines two logical entities located in the access network:
  - ATCF—The (Access Transfer Control Function) is a signalling controller function complementing the signalling and media control roles of the P-CSCF and IMS-AGW functions in the SBC.
  - ATGW—The Access Transfer Gateway is a media anchor point complementing the signalling and media control roles of the P-CSCF and IMS-AGW functions.

Refer to 3GPP specifications for complete element definitions and explanations of the functions they can or must perform. Key functions and considerations that introduce the Oracle USM's operation are briefly discussed below.

As P-CSCF, the Oracle USM adds traffic policing, UA functions for special SIP signaling handling, CDR generation, topology hiding, policy enforcement, and hosted NAT traversal to the 3GPP-defined P-CSCF functions. As such, the Oracle USM can reside at the network's edge performing functions that may otherwise be performed by an SBC.

As I-CSCF, the Oracle USM complies with 3GPP standards to perform the interrogating function and locate the proper S-CSCF for a given session.

As S-CSCF, the near the subscriber that facilitate rapid and predictable handover from LTE to circuit 2G/3G networks and update the VCC application server after the access transfer. They facilitate rapid and predictable handover from LTE to circuit 2G/3G networks and update the VCC application server after the access transfer. The combination of these new functions and improved call flow reduces the signalling hops required to handover the active voice call to the new access network complies with 3GPP standards and Oracle USM to manage sessions. It interacts with the HSS to determine whether any given registration can reside locally, or be managed by another S-CSCF device. It also interacts with the HSS and other infrastructure components to provide applicable services within the context of a given session.

As ATCF and ATGW, Oracle USM provides services near the subscriber that facilitate rapid and predictable handover from LTE to circuit 2G/3G networks and update the VCC application server after the access transfer. The combination of these new functions and improved call flow reduces the signalling hops required to handover the active voice call to the new access network

Oracle USM and OTT

The Oracle USM may be deployed as a SIP registrar in OTT environments. OTT environments may use ENUM resources as subscriber database. Alternatively, they may use local subscriber tables (LST). Both ENUM and LST deployment descriptions and configuration instructions are provided within this guide.

Multiple Control Functions on a Single Device

Most Oracle USM deployments are expected to be homogenous. That is, the Oracle USM would most commonly be operating with other Oracle USMs. In addition, because individual Oracle USMs can perform all CSCF functions, the network administrator can expect traffic patterns that are different from deployments that use discrete components to perform CSCF functions.

For example, it would be common for the subscriber database to allow the S-CSCF within a given Oracle USM to take ownership of a UE seeking services via that same device's P-CSCF. In these cases, the user can expect the effective path and service route between a UE and an S-CSCF to be the same as the path between that UE and the P-CSCF.

IMS design anticipates multiple devices performing CSCF functions. Therefore, the Oracle USM also supports operation across multiple Oracle USMs.

Elements of Oracle USM Configuration

Oracle USM consists of multiple configuration elements. This guide presents these elements, separating them along conceptual category with chapters roughly equating to configuration sequence. This section lists configuration elements, providing the reader with a consolidated picture of overall product configuration.

See the Base Configuration Elements Appendix for minimal configuration setting examples that establish an operable Oracle USM.

USM Configuration Elements

Required initial device configuration elements, explained in the Getting Started chapter, include:

- Boot Parameters
- Management Interfaces
- Device Password
- Default Gateway
- Product licensing/entitlement

Required network and SIP service configuration components, explained in multiple chapters, include:

- Enable SIP-Config—System Configuration Chapter
- Service physical and network interface(s)—System Configuration Chapter
- SIP Interfaces—System Configuration Chapter
- SIP Ports—System Configuration Chapter
- Realms—Realms and Nested Realms Chapter
- ENUM—Routing with Local Policy Chapter

Required IMS configuration elements, explained in the Oracle USM Supporting the IMS Core Chapter, include:

- Subscriber Database
- SIP Registrar
- Authentication Profile
- ENUM for e.164 Translation
- Registration Event
- IMS Access Interface

Common Configuration Elements

Common configuration that may be needed for your deployment includes:

- Session Agents
- ENUM Routing
- Local Routing
- Initial Filter Criteria (iFC)
- 3rd Party Registration Service
- IMS Access Functions
- IMS Interconnect Functions
- Media manager
- Media Steering pools

Common secondary management element configuration includes:

- High Availability (HA)
- CDR Accounting
- SNMP Management

Other Configuration Elements

Configuration elements that are available, but may not be required for your deployment include:

- Assorted SIP Configuration
- Number Translation
- Admission Control and QoS
- DoS and other Security Configuration
- Diameter Policy Configuration
- Transcoding
- Local Routing Configuration
- Realm-based media policy and controls
- Traffic Monitoring

See the Appendix on Base USM Configuration Elements for a list of configuration setting examples that bring your system to a minimally operational state in an IMS environment. Change addressing and other infrastructure-dependent setting examples to match that of your environment.

High Availability

Oracle USMs are deployed in pairs to deliver continuous high availability (HA) for interactive communication services. The HA design guarantees that no stable calls are dropped in the event of any single point failure. Furthermore, the Oracle USM HA design provides for full media, registration, call and service state to be shared across an HA node. The solution uses a VRRP-like design, where the two systems share a virtual MAC address and virtual IPv4 address for seamless switchovers.

In the HA pair, one Oracle USM is the primary system, and is used to process signaling and media traffic. The backup system remains fully synchronized with the primary system's session status. The primary system continuously monitors itself for connectivity and internal process health. If it detects service-disrupting conditions or degraded service levels, it will alert the backup Oracle USM to become the active system.

2

Getting Started with the Oracle USM

Prior to configuring your Oracle USM for service, we recommend that you review the information and procedures in this chapter.

This chapter offers information that will help you:

- Review hardware installation procedures
- Connect to your Oracle USM using a console connection, Telnet, or SSH (secure shell)
- Become familiar with the Oracle USM's boot parameters and how to change them if needed
- Understand and configure Oracle USM entitlements
- Load and activate a Oracle USM software image
- Choose a configuration mechanism: ALCI, external management application, or ACP/XML
- Enable RADIUS authentication
- Customize your login banner

Oracle CSM and Oracle USM product installation begin with the same software installation packages. The specific product deployment is based on the hardware platform over which the software is installed.

Oracle USM is deployed on specific hardware platforms, not generic COTS computer hardware. The software detects installation platform upon startup. If the hardware is on Acme Packet 4000 or 6000 series hardware, the software configures itself as Oracle USM instead of Oracle CSM. From that point on, the user has access to only Oracle USM commands and configuration elements. Oracle USM deployments require that the installation team refer to Acme Packet hardware documentation for installation procedures.

Oracle USM Installation and Startup

After you have completed the hardware installation procedures outlined in the *Acme Packet 4000 Hardware Installation Guide* or *Acme Packet 6000 Hardware Installation Guide*, you are ready to establish a connection to your Oracle USM. Then you can load the Oracle USM software image you want to use and establish basic operating parameters.

Hardware Installation Process

Installing the Oracle USM hardware in a rack requires the following process.

1. Unpack the Oracle USM hardware.
2. Install the Oracle USM hardware into the rack.
3. Install the power supplies.
4. Install the fan modules.
5. Install the physical interface cards.

6. Cable the Oracle USM hardware.

> **Note:**
>
> Complete installation procedures fully and note the safety warnings to prevent physical harm to yourself and damage to the Oracle USM hardware.

For more information, see the hardware documentation.

Connecting to Your Oracle USM

You can connect to your Oracle USM either through a direct console connection, or by creating a remote Telnet or SSH session. Both of these access methods provide you with the full range of configuration, monitoring, and management options.

> **Note:**
>
> By default, Telnet and SFTP connections to your Oracle USM are enabled.

Create a Console Connection

Using a serial connection, you can connect your laptop or PC directly to the Acme Packet hardware. If you use a laptop, you must take appropriate steps to ensure grounding.

One end of the cable plugs into your terminal, and the other end plugs into the RJ-45 Console port on the NIU (or management ports area on the Acme Packet 6300).

To make a console connection to your hardware:

1. Set the connection parameters for your terminal to the default boot settings:
   - Baud rate: 115,200 bits/second
   - Data bits: 8
   - Parity: No
   - Stop bit: 1
   - Flow control: None
2. Connect a serial cable to between your PC and the hardware's console port.
3. Apply power to the hardware.
4. Enter the appropriate password information when prompted to log into User mode of the ACLI.

   You can set the amount of time it takes for your console connection to time out by setting the **console-timeout** parameter in the system configuration. If your connection times out, the login sequence appears again and prompts you for your passwords. The default for this field is 0, which means that no time-out is being enforced.

Incoming Telnet Connections and Time-outs

You can Telnet to your Oracle USM. Using remote Telnet access, you can provision the Oracle USM remotely through the management IP interface.

The Oracle USM, when running on Acme Packet platforms can support up to 5 concurrent Telnet sessions. When running on other platform types, only 4 concurrent Telnet sessions are available. In both cases, only one Telnet session may be in configuration mode at a time.

> **Note:**
>
> Telnet does not offer a secure method of sending passwords. Using Telnet, passwords are sent in clear text across the network.

To Telnet to your Oracle USM, you need to know the IP address of its administrative interface (wancom0/eth0). The wancom0/eth0 IP address of your Oracle USM is found by checking the **inet on ethernet** value in the boot parameters or visible from the front panel display.

You can manage incoming Telnet connections from the ACLI:

- To set a time-out due to inactivity, use the **telnet-timeout** parameter in the system configuration. You can set the number of seconds that elapse before the Telnet connection or SSH connection is terminated. The default for this field is 0, which means that no time-out is being enforced.
- To view the users who are currently logged into the system, use the ACLI **show users** command. You can see the ID, timestamp, connection source, and privilege level for active connections.
- From Superuser mode in the ACLI, you can terminate the connections of other users in order to free up connections. Use the **kill user** command with the corresponding connection ID.
- From Superuser mode in the ACLI, you can globally enable and disable Telnet connections:
  - Telnet service is enabled by default unless explicitly disabled as shipped.
  - To disable Telnet, type the **management disable telnet** command at the Superuser prompt and reboot your system. The Oracle USM then refuses any attempts at Telnet connections. If you want to restart Telnet service, type **management enable telnet**.
- If you reboot your Oracle USM from a Telnet session, you lose IP access and therefore your connection.

SSH Remote Connections

For increased security, you can connect to your Oracle USM using SSH. An SSH client is required for this type of connection.

The Oracle USM supports five concurrent SSH and/or SFTP sessions.

There are two ways to use SSH to connect to your Oracle USM. The first works the way a Telnet connection works, except that authentication takes place before the connection to the Oracle USM is made. The second requires that you set an additional password.

1. To initiate an SSH connection to the Oracle USM without specifying users and SSH user passwords:
   a. Open your SSH client (with an open source client, etc.).
   b. At the prompt in the SSH client, type the **ssh** command, a Space, the IPv4 address of your Oracle USM, and then press Enter.

      The SSH client prompts you for a password before connecting to the Oracle USM. Enter the Oracle USM's User mode password. After it is authenticated, an SSH session is initiated and you can continue with tasks in User mode or enable Superuser mode.

      > **Note:**
      >
      > You can also create connections to the Oracle USM using additional username and password options.

2. To initiate an SSH connection to the Oracle USM with an SSH username and password:
   a. In the ACLI at the Superuser prompt, type the **ssh-password** and press Enter. Enter the name of the user you want to establish. Then enter a password for that user when prompted. Passwords do not appear on your screen.

      ```
      ORACLE# ssh-password
      SSH username [saved]: MJones
      Enter new password: 95X-SD
      Enter new password again: 95X-SD
      ```

      After you configure ssh-password, the SSH login accepts the username and password you set, as well as the default SSH/SFTP usernames: User and admin.
   b. Configure your SSH client to connect to your Oracle USM's management IPv4 address using the username you just created. The standard version of this command would be:

      ```
      ssh -l MJones 10.0.1.57
      ```
   c. Enter the SSH password you set in the ACLI.

      ```
      MJones@10.0.2.54 password: 95X-SD
      ```
   d. Enter your User password to work in User mode on the Oracle USM. Enable Superuser mode and enter your password to work in Superuser mode.
   e. A Telnet session window opens and you can enter your password to use the ACLI.

System Boot

When your Oracle USM boots, the following information about the tasks and settings for the system appear in your terminal window.

- System boot parameters
- From what location the software image is being loaded: an external device or internal flash memory
- Requisite tasks that the system is starting
- Log information: established levels and where logs are being sent

- Any errors that might occur during the loading process

After the loading process is complete, the ACLI login prompt appears.

Oracle USM Boot Parameters

Boot parameters specify the information that your Oracle USM uses at boot time when it prepares to run applications. The Oracle USM's boot parameters:

- Allow you to set the IP address for the management interface (wancom0).
- Allow you to set a system prompt. The target name parameter also specifies the title name displayed in your web browser and SNMP device name parameters.
- Determine the software image to boot and from where the system boots that image.
- Sets up the username and password for network booting from an external FTP server.

In addition to providing details about the Oracle USM's boot parameters, this section explains how to view, edit, and implement them.

When displaying the boot parameters, your screen shows a help menu and the first boot parameter (boot device). Press Enter to continue down the list of boot parameters.

Sample Oracle USM Boot Parameters

The full set of Oracle USM boot parameters appears like the ones in this sample:

```
ORACLE(configure)# bootparam
'.' = clear field;  '-' = go to previous field;  ^D = quit
boot device             : eth0
processor number     : 0
host name     : acmepacket8
file name     : /boot/nnSC600.gz
inet on ethernet (e)    : 10.0.1.57:ffff0000
inet on backplane (b)    : 0.0.0.0
host inet (h)            : 10.0.1.5
gateway inet (g)         : 10.0.0.1
user (u)                 : user
ftp password (pw)        : password
flags (f)                : 0x08
target name (tn)         : acmesystem
startup script (s)       : 0
other (o)                :
NOTE: These changed parameters will not go into effect until reboot. Also, be
aware that some boot parameters may also be changed through the PHY and Network
Interface Configurations.
```

Boot Parameter Definitions

The following table defines each of the Oracle USM's boot parameters.

Boot Parameter	Description
boot device	Management interface name and port number of the device from which an image is downloaded (e.g., wancom0 or eth0) from an external device.
processor number	Processor number on the backplane.
host name	Name of the boot host used when booting from an external device.

Boot Parameter	Description
file name	Name of the image file to be booted; can be entered with the filename path. If you are booting from the flash memory, this filename must always match the filename that you designate when you FTP the image from the source to the Oracle USM. When booting from internal flash memory, this filename must start with /boot); for example, /boot/nnECZ710.bz.
inet on ethernet (e)	Internet address of the Oracle USM. This field can have an optional subnet mask in the form inet_adrs:subnet_mask. If DHCP is used to obtain the parameters, lease timing information may also be present. This information takes the form of lease_duration:lease_origin and is appended to the end of the field. In this parameter, the subnet mask ffff0000 = 255.255.0.0. When you use the ACLI acquire-config command, this is the IPv4 address of the Oracle USM from which you will copy a configuration.
inet on backplane (b)	Internet address of the backplane interface, eth0. This parameter can have an optional subnet mask and/or lease timing information, such as e (inet on ethernet) does.
host inet (h)	Internet address of the boot host used when booting from an external device.
gateway inet (g)	Internet address of the gateway to the boot host. Leave this parameter blank if the host is on the same network.
user (u)	FTP username on the boot host.
ftp password (pw)	FTP password for the FTP user on the boot host.
flags (f)	Codes that signal the Oracle USM from where to boot. Also signals the Oracle USM about which file to use in the booting process. This sequence always starts with 0x (these flags are hexadecimal). The most common codes are: 0x08: Means that the system looks at the filename defined in the boot configuration parameters to determine where to boot from and what file to use. If the file name parameter contains /tffsX/filename, then the system boots off the flash memory (see options below). If the file name parameter just contains a filename, then the Oracle USM boots off the external host defined and looks for the filename in the /tftpboot directory on that host. 0x80008: Used for source routing. If your requirements differ from what these flags allow, contact your Oracle customer support representative for further codes.
target name (tn)	Name of the Oracle USM as it appears in the system prompt. For example, ORACLE> or ORACLE#. You need to know the target name if you are setting up an HA node. This name is required to be unique among Oracle USMs in your network. This name can be 64 characters or less.
startup script (s)	For Oracle use only.
other (o)	For Oracle use only.

Boot Parameter Changes

You can access and edit boot parameters by using either the ACLI or by interrupting the system boot process.

> **Note:**
>
> Changes to boot parameters do not go into effect until you reboot the Oracle USM.

Oracle recommends that you use management port 0 (wancom0) as the boot interface, and that your management network is either:

- directly a part of your LAN for management port 0
- accessible through management port 0

Otherwise, your management messages may use an incorrect source address.

Change Boot Parameters from the ACLI

To access and change boot parameters from the ACLI:

1. In Superuser mode, type configure terminal, and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type bootparam, and press Enter. The boot device parameters display.

   ```
   ORACLE(configure)# bootparam
   '.' = clear field;  '-' = go to previous field;  ^D = quit
   boot device          : eth0
   ```

   To navigate through the boot parameters, press Enter and the next parameter appears on the following line.

   You can navigate through the entire list this way. To go back to a previous line, type a hyphen (-) and press Enter. Any value that you enter entirely overwrites the existing value and does not append to it.

3. To change a boot parameter, type the new value that you want to use next to the old value. For example, if you want to change the image you are using, type the new filename next to the old one. You can clear the contents of a parameter by typing a period and then pressing Enter.

   ```
   ORACLE(configure)# bootparam
   '.' = clear field;  '-' = go to previous field;  ^D = quit
   boot device     : eth0
   processor number    : 0
   host name    : goose
   file name    : /boot/nnPCz100.gz /boot/nnPCz200.gz
   ```

   When you have scrolled through all of the boot parameters, the system prompt for the configure terminal branch displays.

   ```
   ORACLE(configure)#
   ```

4. Exit the configure terminal branch.

5. Reboot the Oracle USM for the changes to take effect.

   The ACLI **reboot** and **reboot force** commands initiate a reboot. With the **reboot** command, you must confirm that you want to reboot. With the **reboot force** command, you do not have make this confirmation.

   ```
   ORACLE# reboot force
   ```

The Oracle USM completes the full booting sequence. If necessary, you can stop the auto-boot at countdown to fix any boot parameters.

If you configured boot parameters correctly, the system prompt displays and you can go ahead with configuration, management, or monitoring tasks.

> **Note:**
>
> If you configured the boot parameters incorrectly, the Oracle USM goes into a booting loop and displays an error message.
>
> ```
> Error loading file: errno = 0x226.
> Can't load boot file!!
> ```
>
> Press the space bar to stop the loop. Correct the error in the boot parameter, and reboot the system.

Change Boot Parameters by Interrupting a Boot in Progress

To access and change boot parameters by interrupting a boot in progress:

1. When the Oracle USM is in the process of booting, you can press the space bar on your keyboard to interrupt when you see the following message appear:

   ```
   Press the space bar to stop auto-boot...
   ```

2. After you stop the booting process, you can enter the letter p to display the current parameters, the letter c to change the boot parameters or the @ (at-sign) to continue booting.

   ```
   [Acme Packet Boot]: c
   '.' = clear field;  '-' = go to previous field;  ^D = quit
   boot device          : wancom0
   ```

   To navigate through the boot parameters, press Enter and the next parameter appears on the following line.

   You can navigate through the entire list this way. To go back to a previous line, type a hyphen (-) and press Enter. Any value that you enter entirely overwrites the existing value and does not append to it.

3. To change a boot parameter, type the new value that you want to use next to the old value. For example, if you want to change the image you are using, type the new filename next to the old one.

   ```
   ORACLE(configure)# bootparam
   '.' = clear field;  '-' = go to previous field;  ^D = quit
   boot device     : wancom0
   processor number     : 0
   host name     : goose
   file name     : /code/nnPCz100.bz /code/nnPCz200.bz
   ```

4. After you have scrolled through the complete list of boot parameters, you return to the boot prompt. To reboot with your changes taking effect, type @ (the at-sign), and press Enter.

   ```
   [Acme Packet Boot]: @
   ```

   The Oracle USM completes the full booting sequence, unless there is an error in the boot parameters.

If you have configured boot parameters correctly, the system prompt displays and you can go ahead with configuration, management, or monitoring tasks.

> **Note:**
>
> If you have configured the boot parameters incorrectly, the Oracle USM goes into a booting loop and displays an error message.
>
> ```
> Error loading file: errno = 0x226.
> Can't load boot file!!
> ```
>
> Press the space bar to stop the loop. Correct the error, and reboot your system.

Acme Packet 6000 Boot Parameters

The Acme Packet 6000's boot parameters are slightly different than the Acme Packet 4500's boot parameters. Boot parameters specify the information your Acme Packet 6000 uses at boot time when it prepares to run the software image. The boot parameters:

- Show the Oracle USM's IPv4 address for the management interface (wancom0)
- Allow you to set a system prompt
- Determine the software image and its location
- Configures the external (T)FTP server used for a net boot

Configuring boot parameters has repercussions for the Oracle USM's physical and network interface configurations. When you configure these interfaces, you can set values that might override the boot parameters.

The following is a screen capture of the bootparam configuration list:

```
[Acme Boot]: p
Boot File        : /boot/bzImage-bones64
IP Address       : 172.44.12.89
VLAN             :
Netmask          : 255.255.0.0
Gateway          : 172.44.0.1
Host IP          :
FTP username     :
FTP password     :
Flags            : 0x00000030
Target Name      : ACMEPACKET
Console Device   : COM1
Console Baudrate : 115200
Other            :


[Acme Boot]: ?
 ?                    - print this list
 @                    - boot (load and go)
 p                    - print boot params
 c                    - change boot params
 v                    - print boot logo with version
 r                    - reboot
 s                    - show license information

 Boot flags:
```

```
0x02  - enable kernel debug
0x04  - disable crashdumps
0x08  - disable bootloader countdown
0x10  - enable debug login
0x40  - use DHCP for wancom0
0x80  - use TFTP instead of FTP
```

Boot File — The name and path of the software image you are booting. Include the absolute path for a local boot from the local /boot volume and for a net boot when a path on the FTP server is needed.

IP Address — IP address of wancom0.

VLAN — VLAN of management network over which this address is accessed.

> **Note:**
>
> VLANs over management interfaces are not supported on the Acme Packet 6000

Netmask — Netmask portion of the wancom0 IP Address.

Gateway — Network Gateway that this wancom0 interface uses.

Host IP — IP Address of FTP server to download and execute a software image from.

FTP Username — FTP Server Username.

FTP Password — FTP Server Password.

Flags — Available flags are printed at the end of the bootparam configuration. Only change this under the director of Oracle personnel.

Target Name — Sets the text to display at every system prompt.

Console Device — Set this to COM1

Console Baud Rate — The speed in bits per second which the console port operates at. It operates at 115200 BPS, 8 data bits, no stop bit, parity NONE.

Other — Unused.

Setting Up System Basics

Before configuring and deploying the Oracle USM, you might want to establish some basic attributes such as a system prompt, new User and Superuser passwords, and NTP synchronization.

New System Prompt

The ACLI system prompt is set in the boot parameters. To change it, access the boot parameters and change the **target name** value to make it meaningful within your network. The target name may be up to 38 characters. A value that identifies the system in some way is often helpful.

Security Update for Default / Hard-coded Passwords

The Oracle USM utilizes default or hard-coded passwords as shipped. These are predictable and pose a security risk if not changed promptly. Password setup must be completed through SSH or Console connections. Passwords need to be set when a particular privilege level is accessed and the system determines when the password is the default value.

Upon the first access of the Oracle USM and at each particular privilege level, a check is performed to determine whether default passwords are being used. If this check returns positive, the system requires that new passwords be set. Password changes must be performed through SSH or Console connections.

> **Note:**
>
> Access to the system through the **admin** or **user** account with the factory default password will be successful only one time and solely to reset said password.

Users and Administrators who attempt to access the system through Telnet, a Web Server, SFTP or other unsupported methods prior to a new password being set will not be granted access. Telnet users will see a banner informing them that connections other than SSH and Console access are not being accepted at this time. SFTP or Web Server users will not be authenticated and their connection will immediately terminate.

In the case where the **admin** and/or **user** passwords have been changed from the shipped, default value prior to the first access of that privilege level, the system continues normally.

Since SSH connections are allowed, system users may log in directly as **admin**.

If remote authentication systems are in place, e.g. TACACS/RADIUS servers, the default password check process does not run since local factory default passwords are not used.

Passwords must meet the current password security standards, which consist of 8-64 characters that include 3 of the following:

- Lowercase letters
- Uppercase letters
- Numerals
- Punctuation

For example, at login to admin level privilege when an appropriate license is found:

```
Password: <Enter Default>

*ALERT*

******************************************************

Admin password has not been set. Please change password now.

******************************************************

* Only alphabetic (upper or lower case), numeric and punctuation
```

```
* characters are allowed in the password. *

* Password must be 8 - 64 characters,

* and have 3 of the 4 following character classes :

* - lower case alpha

* - upper case alpha

* - numerals

* - punctuation

Enter new password: <NEW PASSWORD>

Confirm password: <NEW PASSWORD>

Password is acceptable.

*Set password successfully.
```

The Admin user is then logged in with the new password.

There is no impact on call processing while the default passwords are still in place. Should you encounter a problem with the procedure, contact the Technical Support Center.

Using the Oracle USM Image

The Oracle USM arrives with the most recent, manufacturing-approved run-time image installed on the flash memory. If you want to use this image, you can install the Oracle USM as specified in the *Acme Packet Hardware Installation Guide* , establish a connection to the Oracle USM, and begin to configure it. On boot up, the system displays information about certain configurations not being present. You can dismiss these displays and begin configuring the Oracle USM.

If you want to use an image other than the one installed on the Oracle USM when it arrives, you can use the information in this section to obtain and install the image.

Software Images (USM/CSM)

USM/CSM software image filenames for S-CZ7.3.5 and higher end with the .bz extension. They should be placed in the /boot volume.

Obtaining a New Image

You can download a software image onto the Oracle USM platform from the following sources.

- Obtain an image from the SFTP site and directory where you or your Oracle customer support representative placed the image. For example, this may be a special server that you use expressly for images and backups.
- Obtain an image from your Oracle customer support representative, who will transfer it to your system.

Regardless of the source, use SFTP to copy the image from the source to the Oracle USM.

Copy an Image to the Oracle USM using SFTP

The /boot directory on the Oracle USM has 32mb available, and operating system files of approximately 9mb each. Oracle recommends storing no more than two images at a time in this location. One of these should be the latest version. The /boot directory is used for the on-board system flash memory. If you do not put the image in this directory, the Oracle USM will not find it.

To copy an image on your Oracle USM using SFTP:

1. Go to the directory where the image is located.
2. Check the IP address of the Oracle USM's management port (wancom0). (You might think of this as a management address since it is used in the management of your Oracle USM.)
3. Create the connection to the Oracle USM.

   There is a wide variety of methods to establish SFTP access to your system. For example, Linux systems allow SFTP operation from a terminal. For a windows system, there are many GUI applications that provide SFTP.

   Using your SFTP application, start an SFTP session to the IPv4 address of the Oracle USM management port (wancom0). An SFTP username and SFTP password is required to start the session. The username is always user, and the password is the your User mode login password.
4. Set your SFTP application to copy the image to the **/boot** folder using binary transfer mode.
5. Invoke and confirm your SFTP file transfer to the device.
6. Boot the Oracle USM using the image you just transferred.

   In the ACLI, change any boot configuration parameters that need to be changed. It is especially important to change the filename boot parameter to the filename you used during the SFTP process. Otherwise, your system will not boot properly.

   Alternatively, from the console you can reboot to access the boot prompt and then configure boot parameters from there.
7. In the ACLI, execute the **save-config** command in order to save your changes.
8. Reboot the Oracle USM.
9. The Oracle USM runs through its loading processes and returns you to the ACLI logon prompt.

System Image Filename

The system image filename is a name you set for the image. This is also the filename the boot parameters uses when booting your system. This filename must match the filename specified in the boot parameters. When you use it in the boot parameters, it should always start with /boot to signify that the Oracle USM is booting from the /boot directory.

If the filename set in the boot parameters does not point to the image you want sent to the Oracle USM via SFTP, then you could not only fail to load the appropriate image, but you could also load an image from a different directory or one that is obsolete for your purposes. This results in a boot loop condition that you can fix by stopping the countdown, entering the appropriate filename, and rebooting the Oracle USM.

Booting an Image on Your Oracle USM

You can either boot your Oracle USM from the system's local storage or from an external device. Both locations can store images from which the system can boot. This section describes both booting methods.

For boot parameters to go into effect, you must reboot your Oracle USM. Since a reboot stops all call processing, Oracle recommends performing tasks that call for a reboot during off-peak hours. If your Oracle USMs are set up in an HA node, you can perform these tasks on the standby system first.

Booting from Flash Memory

Once you have installed an image, you can boot your Oracle USM from its flash memory. With the exception of testing an image before you install it on the flash memory, this is generally the method you use for booting.

To boot from your Oracle USM flash memory:

1. Confirm that the boot parameters are set up correctly, and make any necessary changes.

   You can check the boot configuration parameters by accessing the **bootparam** command from the configure terminal menu.

   ```
   ORACLE# configure terminal
   ORACLE# bootparam
   ```

2. Change any boot configuration parameters that you need to change. It is especially important to change the **file name** boot configuration parameter. The file name parameter needs to use the /boot value so that the Oracle USM boots from the flash.
3. Reboot your Oracle USM.
4. You are be returned to the ACLI login prompt. To continue with system operations, enter the required password information.

Booting from an External Device

Booting from an external device means that your Oracle USM connects to a server to retrieve the boot image at boot time. Rather than using an image stored on your system's flash memory, it downloads the image from the external device each time it reboots.

When you are testing a new image before putting it on your Oracle USM, you might want to boot from an external device. Ordinarily, you would not want to boot an image on your Oracle USM this way.

To boot an image from an external device:

1. Confirm that the Oracle USM is cabled to the network from which you are booting. This is port 0 on the rear panel of the Oracle USM chassis (wancom0). The image is loaded from the source using FTP.
2. Log into the system you want to mount.
3. On the Oracle USM, configure the information for the boot parameters and confirm the following:
   - boot device—device to which you will FTP

This parameter value must contain the name of the applicable management interface, and then the number of the appropriate 10/100 port. Usually, this value is wancom0.

- file name—name on the host of the file containing the image
  The image file must exist in the home directory of the user on the image source.
- host inet—IPv4 address of the device off of which you are booting
- gateway inet—IPv4 address of the gateway to use if the device from which you are booting is not on the same network as your Oracle USM
- user—username for the FTP account on the boot host
- password—password for the FTP account on the boot host

4. Reboot your Oracle USM.
5. You are returned to the ACLI login prompt. To continue with system operations, enter the required password information.

Provisioning Entitlements

The Oracle USM may be self-licensed at the ACLI. Licensing a system for use consists of setting the system type, then setting the desired entitlements.

> **Note:**
>
> You may continue using the legacy licensing system or you may transition to self-provisioning entitlements. In both cases, ensure that your system's functionality abides by your organization's contractual obligations with Oracle.

Enabling functionality on the Oracle USM is based on installing feature and entitlements. These determine the feature groups that are available for use.

For the Oracle USM, the system provisions a set of default entitlements. You can then choose the user-provisioned entitlements for your deployment.

Entitlements and Entitlement Groups

The following features are entitlements that you may enable on your system:

- Accounting
- Administrative Security with ACP/NNC
- IPv6
- Load Balancing (Session Agent Groups)
- Policy Server, which encompasses External BW Mgmt, External CLF Mgmt, and Ext Policy Server
- QoS
- SIPREC

> **Note:**
>
> If fewer than all of the features that comprise an entitlement group are licensed before setting up entitlements for the first time, the full entitlement group will be enabled after switching to the entitlements system.

Provisioning a New System

An uninitialized Oracle USM already has the product type set, and has a set of default entitlements installed. This allows you to begin operation immediately. A system that has legacy keyed-licenses installed is still considered an uninitialized system.

The **setup entitlements** command allows the user to provision further entitlements.

> **Note:**
>
> The **setup product** command is visible, but not relevant to this release.

Provisioning a System with Existing Keyed Licenses

When changing your Oracle USM's licensing technique from the legacy keyed licenses method to the provisioned entitlements method, be aware of the following:

- After first running **setup entitlements**, your system will be "seeded" with the existing licenses' functionality.
- Once your system has been seeded with its prior, keyed functionality to the provisioned entitlements system, functionality may be changed with the **setup entitlements** command.
- You may notice that there are fewer entitlements than there were keyed licenses; this is normal.
- After setting up provisioned entitlements, the **show features** command will still function to display the previously installed keyed entitlements.

Editing and Viewing Entitlements

If you are not changing the product type, and you are only changing the entitlements, you can edit the existing entitlements with the **setup entitlements** command. Executing this command will first display existing entitlements before giving you the option to modify their settings.

The **show entitlements** command is used to display the currently provisioned entitlements and keyed entitlements. You may also use the **setup entitlements** command and type **d** to display the current entitlements. Additionally, upon first executing the **setup entitlements** command, all provisioned entitlements (excluding keyed entitlements) are displayed on the screen.

Keyed-only entitlements

Certain entitlements can only be enabled by entering a license key in the **system** > **license** configuration element, and not through the **setup entitlements** command. Contact Oracle support about purchasing these licenses and receiving a valid code to enter in your system.

The Keyed-only entitlements used in conjunction with provisioned entitlements:

- Lawful Intercept
- Codecs including EVRC family and AMR family

- Software TLS

> **Note:**
>
> not all of the above keyed only entitlements are available for all platforms.

Setting Entitlements on HA Pairs

When setting up an HA pair, you must provision the same product type and entitlements on each system.

Feature Provisioning

1. Type **setup entitlements** at the ACLI. Currently provisioned features are printed on the screen.
2. Type the number of the feature you wish to configure followed by pressing the **<Enter>** Key. Some features are set as enabled/disabled (provisionable feature), and some features are provisioned with a maximum capacity value (provisionable capacity feature). The command will let you provision these values as appropriate.
3. Type **enabled** or **disabled** to set a provisionable feature , or type an integer value for a provisionable capacity feature. Both input types are followed by pressing the **<Enter>** key.
4. Repeat steps 2 and 3 to configure additional entitlements.
5. Type **d** followed by the **<Enter>** key to review the full range of your choices. Note that disabled entitlements display their state as blank.
6. Type **s** followed by the **<Enter>** key to commit your choice as an entitlement for your system. After saving the value succeeds you will be returned to the ACLI.
7. Reboot your Oracle USM.

> **Note:**
>
> When configuring an HA pair, you must provision the same product type and features on each system.

Keyed Licenses and Provisioned Entitlements Compatibility

The Oracle USM supports keyed licenses and the provisioned entitlements process for provisioning features on the system. Know the following about how licensing mechanisms work with each other.

- You are not required to begin using the provisioned entitlement system as of this release.
- You may continue to obtain keyed licenses and install them as necessary.
- Upon migrating to the provisioned entitlements system, the current range of your installed, keyed licenses will be seeded to the provisioned entitlements system. The system's functionality will remain identical.

- If you upgrade to the provisioned entitlements system, then downgrade the software to a version without the provisioned entitlements system, the pre-existing keyed licenses will still function.
- If you upgrade to the provisioned entitlements system, then change the functionality (such as, adding more SIP sessions or removing a feature set), upon downgrade to a pre-provisioned entitlements image, the new functionality will not be present.

Obtaining a License

If you choose to add functionality to your Oracle USM, each new feature will require its own key. To obtain additional licenses for functions on your Oracle USM, contact your customer support or sales representative directly or at support@acmepacket.com. You can request and purchase a license for the software you want, obtain a key for it, and then activate it on your Oracle USM.

When you obtain licenses, you need to provide Oracle with the serial number of your Oracle USM. You can see the system's serial number by using the ACLI **show version boot** command.

Trial License

Oracle offers a trial license for software components to allow testing a feature before deployment.

A trial license is active for specified period of time. When the trial license expires, the feature stops responding and the system removes the configuration selections. To continue using the feature you must obtain a license. For more information about licensing, contact your Oracle sales representative or technical support representative.

Standalone System Licensing

This section shows you how to add licenses and delete them from standalone Oracle USMs. The process for two systems making up an HA node is different, so follow the procedure relevant to your configuration.

Adding a License to a Standalone System

Once you have obtained a license key, you can add it to your Oracle USM and activate it.

To add and activate a license on your Oracle USM:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type license and press Enter.

   ```
   ORACLE(system)# license
   ORACLE(license)#
   ```

4. Using the **add** command and the key generated by Oracle, add the license to your Oracle USM.

```
ORACLE(license)# add sl25o39pvtqhas4v2r2jc1oaen9e01o21b1dmh3
```

5. You can check that the license has been added by using the ACLI **show** command within the license configuration.

```
ORACLE(license)# show
1: High Availability
2: Accounting
3: SIP
4: H323
5: 250 sessions, ACP
6: QOS
ORACLE(license)#
```

6. To activate your license, type the **activate-config** command and press Enter. The Oracle USM then enables any of the processes that support associated features.

```
ORACLE# activate-config
```

Deleting a License from a Standalone System

You can delete a license from your Oracle USM, including licenses that have not expired. If you want to delete a license that has not expired, you need to confirm the deletion.

To delete a license from the Oracle USM:

1. In Superuser mode, type configure terminal and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **system** and press Enter.

```
ORACLE(configure)# system
ORACLE(system)#
```

3. Type license and press Enter.

```
ORACLE(system)# license
ORACLE(license)#
```

4. Type the **no** command and press Enter. A list of possible licenses to delete appears.

```
ORACLE(license)# no
feature:
1: High Availability
2: Accounting
3: SIP
4: H323
5: 250 sessions, ACP
6: QOS
selection:
ORACLE(license)#
```

5. Type the number corresponding to the license you want to delete and press Enter.

```
selection:6
```

6. If the license has not expired, you are be asked to confirm the deletion.

```
Delete unexpired license [y/n]?: y
ORACLE(license)#
```

When you show the licenses, the one you deleted should no longer appear on the list.

7. To clear the license from the system, type the **activate-config** command and press Enter. The Oracle USM then disables any of the processes that support associated features.

```
ORACLE# activate-config
```

Viewing Licenses

There are two ways to view licenses in the ACLI.

- You can use the **show features** command at the main ACLI user prompt.

```
ORACLE# show features
Total session capacity: 2250
Enabled protocols: SIP, H.323, IWF
Enabled features: ACP
ORACLE#
```

- Within the license menu, use the **show** command to see all licenses with detailed information.

```
ORACLE(license)# show
License #1: 2000 sessions, SIP, ACP
                no expiration
                installed at 12:34:42 APR 01 2005
License #2: H323
                expired at 23:59:59 APR 08 2005
                installed at 12:35:43 APR 01 2005
License #3: 250 sessions, IWF
                expires at 23:59:59 APR 28 2005
                installed at 12:36:44 APR 01 2005
License #4: QOS
                starts at 00:00:00 APR 08 2004
                expires at 23:59:59 OCT 27 2005
                installed at 12:37:45 APR 01 2005
Total session capacity: 2250
ORACLE(license)#
```

High Availability (HA) Pair Licensing

When adding and deleting licenses, you must perform the procedure across both members of an HA pair during the same service window. The licenses on each peer must be identical. Peers with mismatched licenses may exhibit unexpected behavior.

Adding a License to an HA Node

To add a license to both systems in an HA node, you add your licenses to both systems and reboot both systems when they are in standby mode. The process requires that you carefully confirm system synchronization in between various steps.

This procedure uses the designations Oracle USM1 as the original active and Oracle USM2 as the original standby.

To add a license on systems in an HA node:

1. Confirm that Oracle USM1 and Oracle USM2 are synchronized.

   You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 5.

- On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
- On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-current-cfg-version
Current configuration version is 5
ORACLE1#
ORACLE2# display-current-cfg-version
Current configuration version is 5
ORACLE2#
```

- On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-running-cfg-version
Running configuration version is 5
ORACLE1#
ORACLE2# display-running-cfg-version
Running configuration version is 5
ORACLE2#
```

2. Now you can add a license to SBC1. To begin, type **configure terminal** and press Enter.

```
ORACLE1# configure terminal
ORACLE1(configure)#
```

3. Type **system** and press Enter.

```
ORACLE1(configure)# system
ORACLE1(system)#
```

4. Type **license** and press Enter.

```
ORACLE1(system)# license
ORACLE1(license)#
```

5. Using the **add** command and the key generated by Oracle, add the license to your Oracle USM.

```
ORACLE1(license)# add sjkl4i45987p43hh0938hnhjlaie10983
```

6. You can check that the license has been added by using the ACLI **show** command within the license configuration.

```
ORACLE(license)# show
1: High Availability
2: Accounting
3: SIP
4: H323
5: 250 sessions, ACP
6: QOS
ORACLE(license)#
```

7. Repeat typing **exit**, pressing Enter after each entry, until you reach the main Superuser prompt.

```
ORACLE1(license)# exit
ORACLE1(system)# exit
ORACLE1(configure)# exit
ORACLE1#
```

8. Repeat steps 2 through 7 on SBC2.

```
ORACLE2(license)# add [License for SBC2]
```

At the end of this step, licenses are installed and verified on *both* SBC1 and SBC2.

9. Return to SBC1 and type the **save-config** command and press Enter.

```
ORACLE1# save-config
```

10. Type the **activate-config** command and press Enter. The Oracle USM then enables any of the processes that support associated features.

```
ORACLE1# activate-config
```

11. Confirm that Oracle USM1 and Oracle USM2 are synchronized.

You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 6.

- On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
- On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-current-cfg-version
Current configuration version is 6
ORACLE1#
ORACLE2# display-current-cfg-version
Current configuration version is 6
ORACLE2#
```

- On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-running-cfg-version
Running configuration version is 6
ORACLE1#
ORACLE2# display-running-cfg-version
Running configuration version is 6
ORACLE2#
```

12. Execute the ACLI **show health** command to make sure that all processes are synchronized.
13. Return to SBC2 and execute the reboot command.

```
ORACLE2# reboot
```

14. Reconfirm that Oracle USM1 and Oracle USM2 are synchronized.

You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 6.

- On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
- On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-current-cfg-version
Current configuration version is 6
ORACLE1#
```

```
ORACLE2# display-current-cfg-version
Current configuration version is 6
ORACLE2#
```

- On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-running-cfg-version
Running configuration version is 6
ORACLE1#
ORACLE2# display-running-cfg-version
Running configuration version is 6
ORACLE2#
```

15. Trigger a switchover between the two systems in the HA node so the originally standby system assumes the active role. This means that the standby system will transition to active.

```
ORACLE1# notify berpd force
```

16. Wait for Oracle USM2 to transition to the active state. Confirm that it is in the active state by using the ACLI **show health** command.

17. Reconfirm that Oracle USM1 and Oracle USM2 are synchronized.

    You must also make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 6.

    - On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
    - On Oracle USM2, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM1 and be sure that its current configuration version is the same as the one on Oracle USM2.

```
ORACLE2# display-current-cfg-version
Current configuration version is 6
ORACLE2#
ORACLE1# display-current-cfg-version
Current configuration version is 6
ORACLE1#
```

    - On Oracle USM2, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM1 and be sure that its running configuration version is the same as the one on Oracle USM2.

```
ORACLE2# display-running-cfg-version
Running configuration version is 6
ORACLE2#
ORACLE1# display-running-cfg-version
Running configuration version is 6
ORACLE1#
```

18. Return to SBC1 and execute the reboot command.

```
ORACLE1# reboot
```

19. Wait for SBC1 to complete rebooting. When finished, execute the ACLI **show health** command to make sure that all processes are synchronized.

    At this point both SBCs should be synchronized and contain the same license configuration.

20. If desired, trigger a switchover between the two systems in the HA node so the originally active system (SBC1) assumes the active role.

    ```
    ORACLE2# notify berpd force
    ```

    At this point both SBCs should be synchronized and contain the same license configuration.

Deleting a License from an HA Node

To delete a license from both systems in an HA node, you remove your licenses from both systems and reboot both systems when they are in standby mode. The process requires that you carefully confirm system synchronization in between various steps.

> **Note:**
>
> Licenses should be deleted on both nodes during the same service window.

This procedure uses the designations Oracle USM1 as the original active and Oracle USM2 as the original standby.

To delete a license from systems in an HA node:

1. Confirm that Oracle USM1 and Oracle USM2 are synchronized.

   You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 7.

   - On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
   - On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

     ```
     ORACLE1# display-current-cfg-version
     Current configuration version is 7
     ORACLE1#
     ORACLE2# display-current-cfg-version
     Current configuration version is 7
     ORACLE2#
     ```

   - On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

     ```
     ORACLE1# display-running-cfg-version
     Running configuration version is 7
     ORACLE1#
     ORACLE2# display-running-cfg-version
     Running configuration version is 7
     ORACLE2#
     ```

2. Now you can delete a license from SBC1. To begin, type **configure terminal** and press Enter.

   ```
   ORACLE1# configure terminal
   ORACLE1(configure)#
   ```

3. Type **system** and press Enter.

   ```
   ORACLE1(configure)# system
   ```

4. Type **license** and press Enter.

   ```
   ORACLE1(system)# license
   ORACLE1(license)#
   ```

5. Type the **no** command and press Enter. A list of possible license to delete appears.

   ```
   ORACLE(license)# no
   1: High Availability
   2: Accounting
   3: SIP
   4: H323
   5: 250 sessions, ACP
   6: QOS
   selection#
   ```

6. Type the number corresponding to the license you want to delete and press Enter.

   ```
   selection:6
   ```

7. If the license has not expired, you are be asked to confirm the deletion.

   ```
   Delete unexpired license [y/n]?: y
   ORACLE1(license)#
   ```

   When you show the licenses, the one you deleted should no longer appear on the list.

8. Repeat typing **exit**, pressing Enter after each entry, until you reach the main Superuser prompt.

   ```
   ORACLE1(license)# exit
   ORACLE1(system)# exit
   ORACLE1(configure)# exit
   ORACLE1#
   ```

9. Repeat steps 2 through 8 on SBC2.

   ```
   ORACLE(license)# no
   feature:
   1: High Availability
   2: Accounting
   3: SIP
   4: H323
   5: 250 sessions, ACP
   6: QOS
   selection#
   ```

   At the end of this step, licenses are removed and verified on *both* SBC1 and SBC2.

10. Return to SBC1 and type the **save-config** command and press Enter.

    ```
    ORACLE1# save-config
    ```

11. Type the **activate-config** command and press Enter.

    ```
    ORACLE1# activate-config
    ```

12. Confirm that Oracle USM1 and Oracle USM2 are synchronized.

    You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 5.

- On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
- On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-current-cfg-version
Current configuration version is 7
ORACLE1#
ORACLE2# display-current-cfg-version
Current configuration version is 7
ORACLE2#
```

- On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-running-cfg-version
Running configuration version is 7
ORACLE1#
ORACLE2# display-running-cfg-version
Running configuration version is 7
ORACLE2#
```

13. Execute the ACLI **show health** command to make sure that all processes are synchronized.
14. Return to SBC2 and execute the reboot command.

```
ORACLE2# reboot
```

15. Reconfirm that Oracle USM1 and Oracle USM2 are synchronized.

    You must make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 7.

    - On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
    - On Oracle USM1, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM2 and be sure that its current configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-current-cfg-version
Current configuration version is 7
ORACLE1#
ORACLE2# display-current-cfg-version
Current configuration version is 7
ORACLE2#
```

    - On Oracle USM1, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM2 and be sure that its running configuration version is the same as the one on Oracle USM1.

```
ORACLE1# display-running-cfg-version
Running configuration version is 7
ORACLE1#
ORACLE2# display-running-cfg-version
Running configuration version is 7
ORACLE2#
```

16. Trigger a switchover between the two systems in the HA node so the originally standby system assumes the active role. This means that the originally standby system will transition to active.

    ```
    ORACLE1# notify berpd force
    ```

17. Wait for Oracle USM2 to transition to the active state. Confirm that it is in the active state by using the ACLI **show health** command.

18. Reconfirm that Oracle USM1 and Oracle USM2 are synchronized.

    You must also make sure that all of the running and current configurations on Oracle USM1 and Oracle USM2 have the same number. In the examples below, all of the configuration versions are 8.

    - On Oracle USM1 and Oracle USM2, use the ACLI show health command to make sure that all processes are synchronized.
    - On Oracle USM2, show the current configuration version by using the ACLI display-current-cfg-version command. Then use the same command on Oracle USM1 and be sure that its current configuration version is the same as the one on Oracle USM2.

      ```
      ORACLE2# display-current-cfg-version
      Current configuration version is 7
      ORACLE2#
      ORACLE1# display-current-cfg-version
      Current configuration version is 7
      ORACLE1#
      ```

    - On Oracle USM2, show the running configuration version by using the ACLI display-running-cfg-version command. Then use the same command on Oracle USM1 and be sure that its running configuration version is the same as the one on Oracle USM2.

      ```
      ORACLE2# display-current-cfg-version
      Current configuration version is 7
      ORACLE2#
      ORACLE1# display-current-cfg-version
      Current configuration version is 7
      ORACLE1#
      ```

19. Return to SBC1 and execute the reboot command.

    ```
    ORACLE1# reboot
    ```

20. Wait for SBC1 to complete rebooting. When finished, execute the ACLI **show health** command to make sure that all processes are synchronized.

21. If desired, trigger a switchover between the two systems in the HA node so the originally active system (SBC1) assumes the active role.

    ```
    ORACLE2# notify berpd force
    ```

    At this point both SBCs should be synchronized and contain the same license configuration.

RADIUS Authentication

A security feature that extends beyond the designation of ACLI User and Superuser privileges, the User Authentication and Access control feature supports authentication using your RADIUS server(s). In addition, you can set two levels of privilege, one for all privileges and more limited set that is read-only.

User authentication configuration also allows you to use local authentication, localizing security to the Oracle USM ACLI log-in modes. These modes are User and Superuser, each requiring a separate password.

The components involved in the RADIUS-based user authentication architecture are the Oracle USM and your RADIUS server(s). In these roles:

- The Oracle USM restricts access and requires authentication via the RADIUS server; the Oracle USM communicates with the RADIUS server using either port 1812 or 1645, but does not know if the RADIUS server listens on these ports
- Your RADIUS server provides an alternative method for defining Oracle USM users and authenticating them via RADIUS; the RADIUS server supports the VSA called ACME_USER_CLASS, which specifies what kind of user is requesting authentication and what privileges should be granted.
  The Oracle USM also supports the use of the Cisco Systems Inc.™ Cisco-AVPair vendor specific attribute (VSA). This attribute allows for successful administrator login to servers that do not support the Oracle authorization VSA. While using RADIUS-based authentication, the Oracle USM authorizes you to enter Superuser mode locally even when your RADIUS server does not return the ACME_USER_CLASS VSA or the Cisco-AVPair VSA. For this VSA, the Vendor-ID is 1 and the Vendor-Type is 9. The list below shows the values this attribute can return, and the result of each:
  - shell:priv-lvl=15—User automatically logged in as an administrator
  - shell:priv-lvl=1—User logged in at the user level, and not allowed to become an administrator
  - Any other value—User rejected

When RADIUS user authentication is enabled, the Oracle USM communicates with one or more configured RADIUS servers that validates the user and specifies privileges. On the Oracle USM, you configure:

- What type of authentication you want to use on the Oracle USM
- If you are using RADIUS authentication, you set the port from which you want the Oracle USM to send messages
- If you are using RADIUS authentication, you also set the protocol type you want the Oracle USM and RADIUS server to use for secure communication

Although most common set-ups use two RADIUS servers to support this feature, you are allowed to configure up to six. Among other settings for the server, there is a class parameter that specifies whether the Oracle USM should consider a specific server as primary or secondary. As implied by these designation, the primary servers are used first for authentication, and the secondary servers are used as backups. If you configure more than one primary and one secondary server, the Oracle USM will choose servers to which it sends traffic in a round-robin strategy. For example, if you specify three servers are primary, the Oracle USM will round-robin to select a server until it finds an appropriate one; it will do the same for secondary servers.

The VSA attribute assists with enforcement of access levels by containing one of the three following classes:

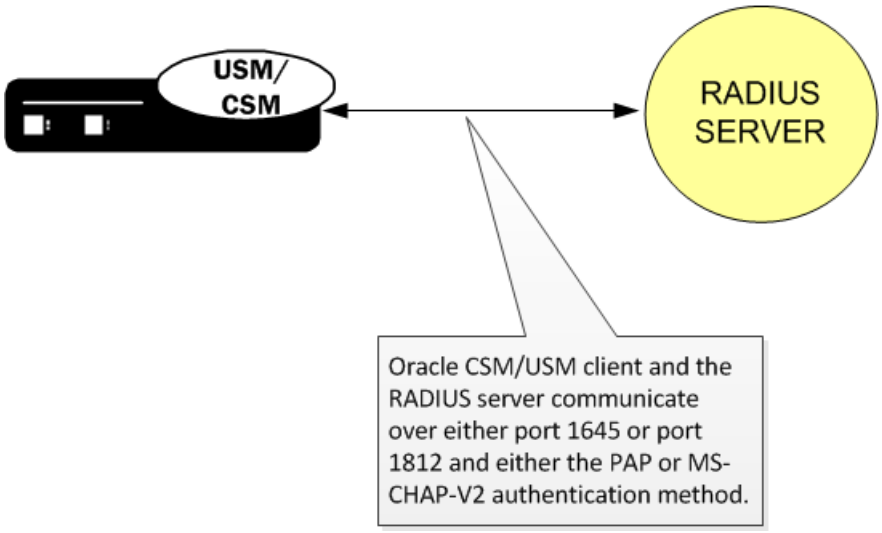
- None—All access denied
- User—Monitoring privileges are granted; your user prompt will resemble ORACLE>
- Admin—All privileges are granted (monitoring, configuration, etc.); your user prompt will resemble ORACLE#

Once it has selected a RADIUS server, the Oracle USM initiates communication and proceeds with the authentication process. The authentication process between the Oracle USM and the RADIUS server takes place uses one of three methods, all of which are defined by RFCs:

Protocol	RFC
PAP (Password Authentication Protocol)	B. Lloyd and W. Simpson, PPP Authentication Protocols, RFC 1334, October 1992
CHAP (Challenge Handshake Authentication Protocol)	B. Lloyd and W. Simpson, PPP Authentication Protocols, RFC 1334, October 1992 W. Simpson, PPP Challenge Handshake Authentication Protocol (CHAP), RFC 1994, August 1996
MS-CHAP-V2	G. Zorn, Microsoft PPP CHAP Extensions, Version 2, RFC 2759, January 2000

Note:

MS-CHAP-V2 support includes authentication only; password exchange is not supported or allowed on the Oracle USM.



PAP Handshake

For PAP, user credentials are sent to the RADIUS server include the user name and password attribute. The value of the User-Password attribute is calculated as specified in RFC 2865.

PAP Client Request Example

```
Radius Protocol
Code: Access Request (1)
  Packet identifier: 0x4 (4)
  Length: 61
  Authenticator: 0x0000708D00002C5900002EB600003F37
  Attribute value pairs
    t:User Name(1) l:11, value:"TESTUSER1"
      User-Name: TESTUSER1
```

```
t:User Password (2) l:18, value:739B3A0F25094E4B3CDA18AB69EB9E4
t:NAS IP Address(4) l:6, value:168.192.68.8
  Nas IP Address: 168.192.68.8(168.192.68.8)
t:NAS Port(5) l:6, value:118751232
```

PAP RADIUS Response

```
Radius Protocol
  Code: Access Accept (2)
  Packet identifier: 0x4 (4)
  Length: 20
  Authenticator: 0x36BD589C1577FD11E8C3B5BB223748
```

CHAP Handshake

When the authentication mode is CHAP, the user credentials sent to the RADIUS server include “username,” “CHAP-Password,” and “CHAP-Challenge.” The “CHAP-Password” credential uses MD-5 one way. This is calculated over this series of the following values, in this order: challenge-id (which for the Oracle USM is always 0), followed by the user password, and then the challenge (as specified in RFC 1994, section 4.1).

CHAP Client Request Example

```
Radius Protocol
  Code: Access Request (1)
  Packet identifier: 0x5 (5)
  Length: 80
  Authenticator: 0x0000396C000079860000312A00006558
  Attribute value pairs
    t:User Name(1) l:11, value:"TESTUSER1"
      User-Name: TESTUSER1
    t:CHAP Password (3) l:19, value:003D4B1645554E881231ED7A137DD54FBF
    t:CHAP Challenge (60) l:18, value: 000396C000079860000312A00006558
    t:NAS IP Address(4) l:6, value:168.192.68.8
      Nas IP Address: 168.192.68.8(168.192.68.8)
    t:NAS Port(5) l:6, value:118751232
```

CHAP RADIUS Response

```
Radius Protocol
  Code: Access Accept (2)
  Packet identifier: 0x4 (4)
  Length: 20
  Authenticator: 0x3BE89EED1B43D91D80EB2562E9D65392
```

MS-CHAP-v2 Handshake

When the authentication method is MS-CHAP-v2, the user credentials sent to the RADIUS server in the Access-Request packet are:

- username
- MS-CHAP2-Response—Specified in RFC 2548, Microsoft vendor-specific RADIUS attributes
- MS-CHAP2-Challenge—Serves as a challenge to the RADIUS server

If the RADIUS authentication is successful, the Access-Accept packet from the RADIUS server must include an MS-CHAP2-Success attribute calculated using the MS-CHAP-Challenge attribute included in the Access-Request. The calculation of MS-CHAP2-Success must be carried out as specified in RFC 2759. The Oracle USM verifies that the MS-CHAP2-Success attribute matches with the calculated value. If the values do not match, the authentication is treated as a failure.

MS-CHAP-v2 Client Request Example

Some values have been abbreviated.

```
Radius Protocol
  Code: Access Request (1)
  Packet identifier: 0x5 (5)
  Length: 80
  Authenticator: 0x0000024C000046B30000339F00000B78
  Attribute value pairs
    t:User Name(1) l:11, value:"TESTUSER1"
      User-Name: TESTUSER1
    t:Vendor Specific(26) l:24, vendor:Microsoft(311)
    t:MS CHAP Challenge(11) l:18, value:0000024C000046B30000339F00000B78
    t:Vendor Specific(26) l:58, vendor:Microsoft(311)
    t:MS CHAP2 Response(25) l:52, value:00000000024C000046B30000339F00000B78...
    t:NAS IP Address(4) l:6, value:168.192.68.8
      Nas IP Address: 168.192.68.8(168.192.68.8)
    t:NAS Port(5) l:6, value:118751232
```

MS-CHAP-v2 RADIUS Response

```
Radius Protocol
  Code: Access Accept (2)
  Packet identifier: 0x6 (6)
  Length: 179
  Authenticator: 0xECB4E59515AD64A2D21FC6D5F14D0CC0
  Attribute value pairs
    t:Vendor Specific(26) l:51, vendor:Microsoft(311)
      t:MS CHAP Success(11) l:45, value:003533s33d3845443532443135453846313...
    t:Vendor Specific(26) l:42, vendor:Microsoft(311)
      t:MS MPPE Recv Key(17) l:36, value:96C6325D22513CED178F770093F149CBBA...
    t:Vendor Specific(26) l:42, vendor:Microsoft(311)
      t:MS MPPE Send Key(16) l:36, value:9EC9316DBFA701FF0499D36A1032678143...
    t:Vendor Specific(26) l:12, vendor:Microsoft(311)
      t:MS MPPE Encryption Policy(7) l:6, value:00000001
    t:Vendor Specific(26) l:12, vendor:Microsoft(311)
      t:MS MPPE Encryption Type(8) l:6, value:00000006
```

Management Protocol Behavior

When you use local authentication, management protocols behave the same way that they do when you are not using RADIUS servers. When you are using RADIUS servers for authentication, management protocols behave as described in this section.

- Telnet—The “user” or admin accounts are authenticated locally, not via the RADIUS server. For all other accounts, the configured RADIUS servers are used for authentication. If authentication is successful, the user is granted privileges depending on the ACME_USER_CLASS VSA attribute.

- FTP—The “user” or admin accounts are authenticated locally, not via the RADIUS server. For all other accounts, the configured RADIUS servers are used for authentication.
- SSH in pass-through mode—When SSH is in pass through mode, the Oracle USM behave the same way that it does for Telnet.
- SSH in non-pass-through mode—When you create an SSH account on the Oracle USM, you are asked to supply a user name and password. Once local authentication succeeds, you are prompted for the ACLI user name and password. If your user ACLI name is user, then you are authenticated locally. Otherwise, you are authenticated using the RADIUS server. If RADIUS authentication is successful, the privileges you are granted depend on the ACME_USER_CLASS VSA attribute.
- SFTP in pass-through mode—If you do not configure an SSH account on the Oracle USM, the RADIUS server is contacted for authentication for any user that does not have the user name user. The Oracle USM uses local authentication if the user name is user.
- SFTP in non-pass-through mode—The “user” or admin accounts are authenticated locally, not via the RADIUS server. For all other accounts, the configured RADIUS servers are used for authentication.

RADIUS Authentication Configuration

To enable RADIUS authentication and user access on your Oracle USM, you need to configure global parameters for the feature and then configure the RADIUS servers that you want to use.

Global Authentication Settings

To configure the global authentication settings:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ```

3. Type **authentication** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(security)# authentication
   ORACLE(authentication)#
   ```

   From here, you can view the entire menu for the authentication configuration by typing a **?**. You can set global parameters for authentication. You can also configure individual RADIUS servers; instructions for configuring RADIUS server appear in the next section.

4. **type**—Set the type of user authentication you want to use on this Oracle USM. The default value is **local**. The valid values are:

   - local | radius

5. **protocol**—If you are using RADIUS user authentication, set the protocol type to use with your RADIUS server(s). The default is **pap**. The valid values are:

   - pap | chap | mschapv2

6. **source-port**—Set the number of the port you want to use from message sent from the Oracle USM to the RADIUS server. The default value is 1812. The valid values are:

   - 1645 | 1812

7. **allow-local-authorization**—Set this parameter to **enabled** if you want the Oracle USM to authorize users to enter Superuser (administrative) mode locally even when your RADIUS server does not return the ACME_USER_CLASS VSA or the Cisco-AVPair VSA. The default for this parameter is **disabled**.

RADIUS Server Settings

The parameters you set for individual RADIUS servers identify the RADIUS server, establish a password common to the Oracle USM and the server, and establish trying times.

Setting the class and the authentication methods for the RADIUS servers can determine how and when they are used in the authentication process.

To configure a RADIUS server to use for authentication:

1. Access the RADIUS server submenu from the main authentication configuration:

```
ORACLE(authentication)# radius-servers
ORACLE(radius-servers)#
```

2. **address**—Set the remote IP address for the RADIUS server. There is no default value, and you are required to configure this address.
3. **port**—Set the port at the remote IP address for the RADIUS server. The default port is set to **1812**. The valid values are:
   - 1645 | 1812
4. **state**—Set the state of the RADIUS server. Enable this parameter to use this RADIUS server to authenticate users. The default value is **enabled**. The valid values are:
   - enabled | disabled
5. **secret**—Set the password that the RADIUS server and the Oracle USM share. This password is transmitted between the two when the request for authentication is initiated; this ensures that the RADIUS server is communicating with the correct client.
6. **nas-id**—Set the NAS ID for the RADIUS server. There is no default for this parameter.
7. **retry-limit**—Set the number of times that you want the Oracle USM to retry for authentication information from this RADIUS server. The default value is **3**. The valid range is:
   - Minimum—1
   - Maximum—5

     If the RADIUS server does not respond within this number of tries, the Oracle USM marks is as dead.
8. **retry-time**—Set the amount of time (in seconds) that you want theOracle USM to wait before retrying for authentication from this RADIUS server. The default value is **5**. The valid range is:
   - Minimum—5
   - Maximum—10
9. **dead-time**—Set the amount of time in seconds before the Oracle USM retries a RADIUS server that it has designated as dead because that server did not respond within the maximum number of retries. The default is **10**. The valid range is:
   - Minimum—10
   - Maximum—10000

10. **maximum-sessions**—Set the maximum number of outstanding sessions for this RADIUS server. The default value is **255**. The valid range is:
    - Minimum—1
    - Maximum—255
11. **class**—Set the class of this RADIUS server as either primary or secondary. A connection to the primary server is tried before a connection to the secondary server is tried. The default value is **primary**. Valid values are:
    - primary | secondary

      The Oracle USM tries to initiate contact with primary RADIUS servers first, and then tries the secondary servers if it cannot reach any of the primary ones.

      If you configure more than one RADIUS server as primary, the Oracle USM chooses the one with which it communicates using a round-robin strategy. The same strategy applies to the selection of secondary servers if there is more than one.
12. **authentication-methods**—Set the authentication method you want the Oracle USM to use with this RADIUS server. The default value is **pap**. Valid values are:
    - all | pap | chap | mschapv2

      This parameter has a specific relationship to the global protocol parameter for the authentication configuration, and you should exercise care when setting it. If the authentication method that you set for the RADIUS server does not match the global authentication protocol, then the RADIUS server is not used. The Oracle USM simply overlooks it and does not send authentication requests to it. You can enable use of the server by changing the global authentication protocol so that it matches.
13. Save your work and activate your configuration.

TACACS+ AAA

TACACS+ (Terminal Access Controller Access Control System Plus) is a protocol originally developed by Cisco Systems, and made available to the user community by a draft RFC, *TACACS+ Protocol, Version 1.78* (draft-grant-tacacs-02.txt). TACACS+ provides AAA (Authentication, Authorization, and Accounting) services over a secure TCP connection using Port 49.

TACACS+ Introduction

Like DIAMETER and RADIUS, TACACS+ uses a client/server model in which a Network Access Server (NAS) acts in the client role and a TACACS+ equipped device (a daemon in TACACS+ nomenclature) assumes the server role. For purposes of the current implementation, the Oracle Oracle USM functions as the TACACS+ client. Unlike RADIUS, which combines authentication and authorization, TACACS+ provides three distinct applications to provide finer grade access control.

Authentication is the process that confirms a user's purported identity. Authentication is most often based on a simple username/password association, but other, and more secure methods, are becoming more common. The following authentication methods are support by the current implementation: simple password, PAP (Protocol Authentication Protocol), and CHAP (Challenge Handshake Authentication Protocol).

Authorization is the process that confirms user privileges. TACACS+ can provide extremely precise control over access to system resources. In the current implementation, TACACS+ controls access to system administrative functions.

TACACS+ provides secure communication between the client and daemon by encrypting all packets. Encryption is based on a shared-secret, a string value known only to the client and daemon. Packets are encrypted in their entirety, save for a common TACACS+ header.

The cleartext header contains, among other fields, a version number, a sequence number. and a session ID. Using a methodology described in Section 5 of the TACACS+ draft RFC, the sender encrypts outbound cleartext messages by repetitively running the MD5 hash algorithm over the concatenation of the session ID, shared-secret, version number, and sequence number values, eventually deriving a virtual one-time-pad of the same length as the message body. The sender encrypts the cleartext message with an XOR (Exclusive OR) operation, using the cleartext message and virtual one-time-pad as inputs.

The message recipient, who possesses the shared-secret, can readily obtain the version number, sequence number, session ID, and message length from the cleartext header. Consequently, the recipient employs the same methodology to derive a virtual one-time-pad identical to that derived by the sender. The recipient decrypts the encrypted message with an XOR operation, using the encrypted message and virtual one-time-pad as inputs.

Details on TACACS+ are available in the *ACLI Configuration Guide*.

The TACACS+ implementation is based upon the following internet draft.

draft-grant-tacacs-02.txt, *The TACACS+ Protocol Version 1.78*

Other relevant documents include

RFC 1321, *The MD-5 Message Digest Algorithm*

RFC 1334, *PPP Authentication Protocols* .

RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*

TACACS+ Administrative Security

Oracle USMs use either the RADIUS (Remote Authentication Dial-In User Service) or the TACACS+ (Terminal Access Control Access Control System Plus) protocol for centralized access control administration; however, prior to this release, you could connect to the TACACS + server only from the system's media interfaces. This feature implements TACACS+ authorization (user permissions management on a command basis), authentication (user management), and accounting on management interfaces.

TACACS+ Authentication

The Oracle USM uses TACACS+ authentication services solely for the authentication of user accounts. Administrative users must be authenticated locally by the Oracle USM.

The current TACACS+ implementation supports three types of user authentication: simple password (referred to as ascii by TACACS+), PAP, and CHAP.

ascii Login

ascii login is analogous to logging into a standard PC. The initiating peer is prompted for a username, and, after responding, is then prompted for a password.

PAP Login

PAP is defined in RFC 1334, *PPP Authentication Protocols*. This protocol offers minimal security in that passwords are transmitted as unprotected cleartext. PAP login differs from ascii login in that the username and password are transmitted to the authenticating peer in a single authentication packet, as opposed to the two-step prompting process used in ascii login.

CHAP Login

CHAP is defined in RFC 1994, *PPP Challenge Handshake Authentication Protocol.* CHAP is a more secure than PAP in that it is based on a shared-secret (known only to the communicating peers), and therefore avoids the transmission of cleartext authentication credentials. CHAP operations can be summarized as follows.

After a login attempt, the initiator is tested by the authenticator who responds with a packet containing a challenge value — an octet stream with a recommended length of 16 octets or more. Receiving the challenge, the initiator concatenates an 8-bit identifier (carried within the challenge packet header), the shared-secret, and the challenge value, and uses the shared-secret to compute an MD-5 hash over the concatenated string. The initiator returns the hash value to the authenticator, who performs the same hash calculation, and compares results. If the hash values match, authentication succeeds; if hash values differ, authentication fails.

Authentication Message Exchange

All TACACS+ authentication packets consist of a common header and a message body. Authentication packets are of three types: START, CONTINUE, and REPLY.

START and CONTINUE packets are always sent by the Oracle USM, the TACACS+ client. START packets initiate an authentication session, while CONTINUE packets provide authentication data requested by the TACACS+ daemon. In response to every client-originated START or CONTINUE, the daemon must respond with a REPLY packet. The REPLY packet contains either a decision (pass or fail), which terminates the authentication session, or a request for additional information needed by the authenticator.

TACACS+ Header

The TACACS+ header format is as follows.

```
+----+----+--------+--------+--------+
|maj |min | type   | seq_no | flags  |
|ver |ver |        |        |        |
+----+----+--------+--------+--------+
| session_id                         |
+------------------------------------+
| length                             |
+------------------------------------+
```

maj ver

This 4-bit field identifies the TACACS+ major protocol version, and must contain a value of 0xC .

min ver

This 4-bit field identifies the TACACS+ minor protocol version, and must contain either a value of 0x0 (identifying TACACS+ minor version 0) or a value of 0x1 . (identifying TACACS

+ minor version 1). Minor versions 0 and 1 differ only in the processing of PAP and CHAP logins.

type

This 8-bit field identifies the TACACS+ AAA service as follows:

0x1 — TACACS+ Authentication

0x2 — TACACS+ Authorization

0x3 — TACACS+ Accounting

sequence-no

This 8-bit field contains the packet sequence for the current session.

The first packet of a TACACS+ session must contain the value 1; each following packet increments the sequence count by 1. As TACACS+ sessions are always initiated by the client, all client-originated packets carry an odd sequence number, and all daemon-originated packets carry an even sequence number. TACACS+ protocol strictures do not allow the sequence_no field to wrap. If the sequence count reaches 255, the session must be stopped and restarted with a new sequence number of 1.

flags

This 8-bit field contains flags as described in Section 3 of the draft RFC; flags are not under user control.

session_id

This 32-bit field contains a random number that identifies the current TACACS+ session — it is used by clients and daemons to correlate TACACS+ requests and responses.

length

This 32-bit field contains the total length of the TACACS+ message, excluding the 12-octet header — in other words, the length of the message body.

Authentication START Packet

The Oracle USM, acting as a TACACS+ client, sends an authentication START packet to the TACACS+ daemon to initiate an authentication session. The daemon must respond with a REPLY packet.

The authentication START packet format is as follows.

```
+---------------------------------+
|           Common Header         |
|                                 |
|         type contains 0x1       |
+--------+--------+--------+------+
|action  |priv_lvl|authen_ |service|
|        |        |type    |       |
|--------+--------+--------+------+
|user_len|port_len|rem_addr|data_len|
|        |        |_len    |       |
+--------+--------+--------+------+
|             user ...            |
+---------------------------------+
|             port ...            |
+---------------------------------+
```

```
|            rem-addr ...           |
+-----------------------------------+
|              data ...             |
+-----------------------------------+
```

action

This 8-bit field contains an enumerated value that identifies the requested authentication action. For the current TACACS+ implementation, this field always contains a value of 0x01 , indicating user login authentication.

priv_lvl

This 8-bit field contains an enumerated value that identifies the privilege level requested by an authenticating user. For the current TACACS+ authentication implementation, this field always contains a value of 0x01 , indicating the user level.

authen-type

This 8-bit field contains an enumerated value that identifies the authentication methodology. Supported values are as follows:

0x01 ASCII — simple login, Oracle USM prompts for username and password

0x02 PAP — as specified in RFC 1334

0x03 CHAP — as specified in RFC 1994

service

This 8-bit field contains an enumerated value that identifies the service requesting the authentication. For the current TACACS+ implementation, this field always contains a value of 0x01 , indicating user login authentication.

user_len

This 8-bit field contains the length of the user field in octets.

port_len

This 8-bit field contains the length of the port field in octets. As the port field is not used in the current TACACS+ authentication implementation, the port_len field always contains a value of 0 as specified in Section 4 of the TACACS+ draft RFC.

rem_addr_len

This 8-bit field contains the length of the rem_addr field in octets. As the rem_addr field is not used in the current TACACS+ authentication implementation, the rem_addr_len field always contains a value of 0 as specified in Section 4 of the TACACS+ draft RFC.

data_len

This 8-bit field contains the length of the data field in octets.

user

This variable length field contains the login name of the user to be authenticated.

port

This variable length field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .

rem_addr

This variable length field contains the location of the user to be authenticated. This field contains the localhost address.

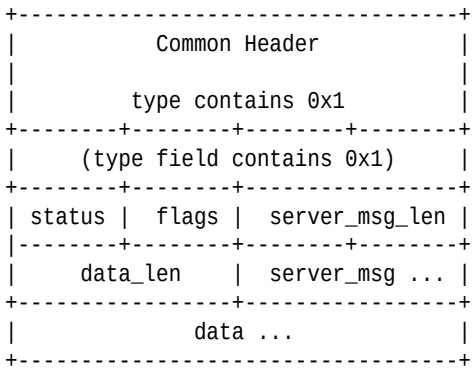
data

This optional variable length field contains miscellaneous data.

Authentication REPLY Packet

The TACACS+ daemon sends an authentication REPLY packet to the Oracle USM in response to a authentication START or authentication CONTINUE packet. Depending on the contents of the status field, the authentication REPLY packet either ends the authentication transaction, or continues the transaction by requesting addition information needed by the authenticator.

The authentication REPLY packet format is as follows.



status

This 16-bit field contains an enumerated value that specifies the current state of the authentication process. Supported values are as follows:

0x01 PASS — the user is authenticated, thus ending the session

0x02 FAIL — the user is rejected, thus ending the session

0x04 GETUSER — daemon request for the user name

0x05 GETPASS — daemon request for the user password

0x06 RESTART — restarts the transaction, possibly because the sequence number has wrapped, or possibly because the requested authentication type is not supported by the daemon

0x07 ERROR — reports an unrecoverable error

flags

This 8-bit field contains various flags that are not under user control.

server_msg_len

This 16-bit field contains the length of the server_msg field in octets. As the server_msg field is not used in REPLY packets sent by the current TACACS+ authentication implementation, the server_msg_len field always contains a value of 0 as specified in Section 4 of the TACACS+ draft RFC.

data_len

This 16-bit field contains the length of the data field in octets. As the data field is not used in REPLY packets sent by the current TACACS+ authentication implementation, the data_len field always contains a value of 0 as specified in Section 4 of the TACACS+ draft RFC.

server_msg

This optional variable length field contains a server message intended for display to the user. The current TACACS+ authentication implementation does not use this field.

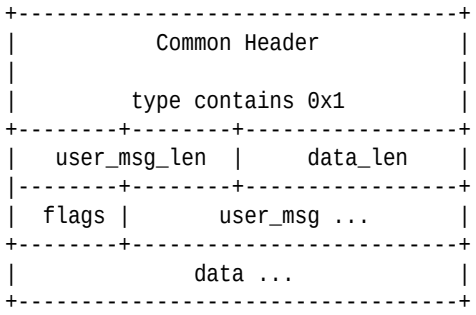
data

This optional variable length field contains data pertinent to the authentication process. The current TACACS+ authentication implementation does not use this field.

Authentication CONTINUE Packet

The Oracle USM, acting as a TACACS+ client, sends an authentication CONTINUE packet to the TACACS+ daemon in response to a REPLY message which requested additional data required by the authenticator.

The authentication CONTINUE packet format is as follows.



user_msg_len

This 16-bit field contains the length of the user_msg field in octets.

data_len

This 16-bit field contains the length of the data field in octets. As the data field is not used in the current TACACS+ authentication implementation, the data field always contains a value of 0 as specified in Section 4 of the TACACS+ draft RFC.

flags

This 8-bit field contains various flags that are not under user control.

user_msg

This variable length field contains a string that responds to an information request contained in a REPLY message.

data

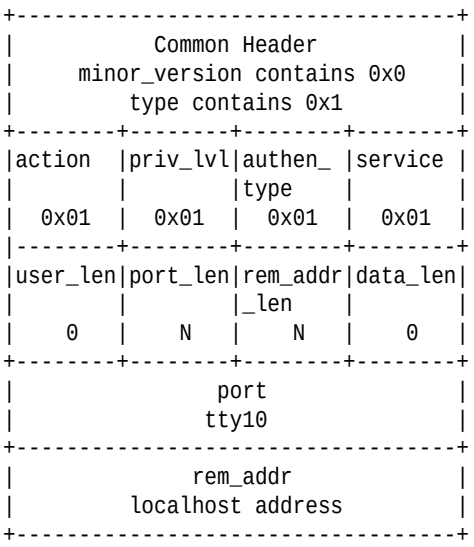
This optional variable length field contains miscellaneous data, often in response to a daemon request. The current TACACS+ authentication implementation does not use the data field in Authentication CONTINUE packets.

Authentication Scenarios

Each of the supported user authentication scenarios is described in terms of packet flow in the following sections.

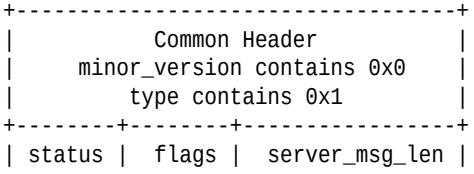
ASCII Authentication

The Oracle USM initiates the authentication with an authentication START packet.



- The action field specifies the requested authentication action — 0x01 for TAC_PLUSAUTHEN_LOGIN (authentication of a user login).
- The priv_lvl field specifies the privilege level requested by the user — 0x01 for TAC_PLUS_PRIV_LVL_USER.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len and data_len fields contain a value of 0 , as required by the TACACS+ protocol.
- The port_len and rem_addr_len fields contain the length, in octets, of the port and rem_addr fields.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.

The TCACS+ daemon returns an authentication REPLY requesting the username.



```
|  0x04  |        |        0        |
|--------+--------+-----------------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies a daemon request — 0x04 for TAC_PLUS_AUTH_STATUS_GETUSER (get username).
- The server_msg_len data_len fields both contain a value of 0 , as required by the TACACS + protocol.

The Oracle USMresponds with an authentication CONTINUE packet.

```
+-----------------------------------+
|           Common Header           |
|     minor_version contains 0x0    |
|         type contains 0x1         |
+-----------------+-----------------+
|   user_msg_len  |     data_len    |
|                 |        0        |
|--------+--------+-----------------+
|  flags |       user_msg ...       |
+--------+--------------------------+
```

- The user_msg_len field contains the length, in octets, of the user_msg field.
- The data_len field contains a value of 0 , as required by the TACACS+ protocol.
- The user_msg field contains the username to be authenticated.

The TCACS+ daemon returns a second authentication REPLY requesting the user password.

```
+-----------------------------------+
|           Common Header           |
|     minor_version contains 0x0    |
|         type contains 0x1         |
+--------+--------+--------+--------+
| status |  flags |  server_msg_len |
|  0x05  |        |        0        |
|--------+--------+--------+--------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies a daemon request — 0x05 for TAC_PLUS_AUTH_STATUS_GETPASS (get user password).
- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.

The Oracle USM responds with a second authentication CONTINUE packet.

```
+-----------------------------------+
|           Common Header           |
|     minor_version contains 0x0    |
|         type contains 0x1         |
+-----------------+-----------------+
|   user_msg_len  |     data_len    |
|                 |        0        |
|--------+--------+--------+--------+
|  flags |       user_msg ...       |
+--------+--------------------------+
```

- The user_msg_len field contains the length, in octets, of the user_msg field.
- The data_len field contains a value of 0 , as required by the TACACS+ protocol.
- The user_msg field contains the user password to be authenticated.
- Other, optional fields are not used.

The TCACS+ daemon returns a third authentication REPLY reporting the authentication result, and terminating the authentication session.

```
+----------------------------------+
|           Common Header          |
|     minor_version contains 0x0   |
|         type contains 0x1        |
+--------+--------+----------------+
| status |  flags |  server_msg_len |
|  0x01  |        |        0       |
|--------+--------+----------------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies the authentication result — 0x01 for TAC_PLUS_AUTH_STATUS_PASS (authorization succeeds), or 0x02 for TAC_PLUS_AUTH_STATUS_FAIL (authorization fails).
- The server_msg_len , and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.

PAP Authentication

The Oracle USM initiates the authentication with an authentication START packet.

```
+----------------------------------+
|           Common Header          |
|     minor_version contains 0x1   |
|         type contains 0x1        |
+--------+--------+--------+-------+
|action  |priv_lvl|authen_ |service |
|        |        |type    |        |
|  0x01  |  0x01  |  0x02  |  0x01  |
|--------+--------+--------+--------+
|user_len|port_len|rem_addr|data_len|
|        |        |_len    |        |
|    N   |    N   |    N   |    N   |
+--------+--------+--------+--------+
|                user               |
+-----------------------------------+
|                port               |
|               tty10               |
+-----------------------------------+
|             rem_addr              |
|         localhost address         |
+-----------------------------------+
|             data ...              |
+-----------------------------------+
```

- The action field specifies the requested authentication action — 0x01 for TAC_PLUSAUTHEN_LOGIN (authentication of a user login).

- The priv_lvl field specifies the privilege level requested by the user — 0x01 for TAC_PLUS_PRIV_LVL_USER.
- The authen_type field specifies the authentication methodology — 0x02 for TAC_PLUS_AUTHEN_TYPE_PAP (PAP login).
- The service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The data_len field contains the length, in octets, of the date field.
- The user field contains the username to be authenticated.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The data field contains the password to be authenticated.

The TCACS+ daemon returns an authentication REPLY reporting the authentication result.

```
+----------------------------------+
|           Common Header          |
|     minor_version contains 0x1   |
|         type contains 0x1        |
+--------+--------+----------------+
| status |  flags |  server_msg_len |
|  0x01  |        |        0       |
|--------+--------+----------------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies the authentication result — 0x01 for TAC_PLUS_AUTH_STATUS_PASS (authorization succeeds), or 0x02 for TAC_PLUS_AUTH_STATUS_FAIL (authorization fails).
- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.
- Other, optional fields are not used.

CHAP Authentication

The Oracle USM initiates the authentication with an authentication START packet.

```
+----------------------------------+
|           Common Header          |
|     minor_version contains 0x1   |
|         type contains 0x1        |
+--------+--------+--------+-------+
|action  |priv_lvl|authen_ |service |
|        |        |type    |        |
|  0x01  |  0x01  |  0x03  |  0x01  |
|--------+--------+--------+--------+
|user_len|port_len|rem_addr|data_len|
```

```
|        |        |_len    |        |
|    N   |    N   |    N   |    N   |
+--------+--------+--------+--------+
|                user                |
+------------------------------------+
|                port                |
|               tty10                |
+------------------------------------+
|              rem_addr              |
|         localhost address          |
+------------------------------------+
|              data ...              |
+------------------------------------+
```

- The action field specifies the requested authentication action — 0x01 for TAC_PLUSAUTHEN_LOGIN (authentication of a user login).
- The priv_lvl field specifies the privilege level requested by the user — 0x01 for TAC_PLUS_PRIV_LVL_USER.
- The authen_type field specifies the authentication methodology — 0x03 for TAC_PLUS_AUTHEN_TYPE_CHAP (CHAP login).
- The service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The data_len field contains the length, in octets, of the date field.
- The user field contains the username to be authenticated.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The data field contains the password to be authenticated.

The TCACS+ daemon returns an authentication REPLY reporting the authentication result.

```
+-----------------------------------+
|           Common Header           |
|     minor_version contains 0x1    |
|         type contains 0x1         |
+--------+--------+-----------------+
| status |  flags |  server_msg_len |
|  0x01  |        |        0        |
|--------+--------+-----------------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies the authentication result — 0x01 for TAC_PLUS_AUTH_STATUS_PASS (authorization succeeds), or 0x02 for TAC_PLUS_AUTH_STATUS_FAIL (authorization fails).
- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.

- Other, optional fields are not used.

TACACS+ Authorization

The Oracle USM uses TACACS+ services to provide administrative authorization. With TACACS+ authorization enabled, each individual ACLI command issued by an admin user is authorized by the TACACS+ authorization service. The Oracle USM replicates each ACLI command in its entirety, sends the command string to the authorization service, and suspends command execution until it receives an authorization response. If TACACS+ grants authorization, the pending command is executed; if authorization is not granted, the Oracle USM does not execute the ACLI command, and displays an appropriate error message.

The daemon's authorization decisions are based on a database lookup. Data base records use regular expressions to associate specific command string with specific users. The construction of such records is beyond the scope of this document.

TACACS+ Authorization Command & Arguments Boundary

Each TACACS+ authorization entry on an ACLI command line comprises the command and its arguments. Currently everything typed as a TACACS+ authorization command by an authenticated admin user, including the arguments, is sent to the TACACS+ server in the command field of the TACACS+ message; the argument field in the TACACS+ message contains no arguments and is set to "cmd-arg=<CR>". This feature adds the new parameter **tacacs-authorization-arg-mode** to the **authentication** configuration element, which enables the TACACS+ authorization command and its arguments to be sent to the TACACS+ server separately.

Authorization Message Exchange

All TACACS+ authorization packets consist of a common header and a message body. Authorization packets are of two types: REQUEST and RESPONSE.

The REQUEST packet, which initiates an authorization session, is always sent by the Oracle USM. Upon receipt of every REQUEST, the daemon must answer with a RESPONSE packet. In the current TACACS+ implementation, the RESPONSE packet must contain an authorization decision (pass or fail). The exchange of a single REQUEST and the corresponding RESPONSE completes the authorization session.

Authorization REQUEST Packet

The Oracle USM, acting as a TACACS+ client, sends an authorization REQUEST packet to the TACACS+ daemon to initiate an authorization session.

The authorization REQUEST packet format is as follows.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x2        |
+--------+--------+--------+--------+
|authen_ |priv_lvl|authen_ |authen- |
|method  |        |type    |service |
|--------+--------+--------+--------+
|user_len|port_len|rem_addr|arg_cnt |
|        |        |_len    |        |
+----+---+--------+--------+--------+
```

```
|arg1_len|arg2_len|  ...   |argN_len|
|        |        |        |        |
+--------+--------+--------+--------+
|              user ...              |
+------------------------------------+
|              port ...              |
+------------------------------------+
|            rem-addr ...            |
+------------------------------------+
|              arg1 ...              |
+------------------------------------+
|              arg2 ...              |
+------------------------------------+
|              argN ...              |
+------------------------------------+
```

authen_method

This 8-bit field contains an enumerated value that identifies the method used to authenticate the authorization subject — that is, an admin user. Because the admin user was authenticated locally by the Oracle USM, this field always contains a value of 0x05 , indicating authentication by the requesting client.

priv_lvl

This 8-bit field contains an enumerated value that identifies the privilege level associated with the authorization subject. For the current TACACS+ authorization implementation, this field always contains a value of 0x00 .

authen-type

This 8-bit field contains an enumerated value that identifies the methodology. used to authenticate the authorization subject. Because the admin user was authenticated with a simple username/password exchange, this field always contains a value of 0x01 , indicating ascii login.

authen_service

This 8-bit field contains an enumerated value that identifies the service that requested authentication. Because an admin user is authenticated with a simple username/password exchange, this field always contains a value of 0x01 , the login service.

user_len

This 8-bit field contains an integer that specifies the length, in octets, of the user field.

port_len

This 8-bit field contains an integer that specifies the length, in octets, of the port field.

rem_addr_len

This 8-bit field contains an integer that specifies the length, in octets, of the rem_addr field.

arg_cnt

This 8-bit field contains an integer that specifies the number or arguments contained with the REQUEST. Given the design of the current TACACS+ implementation, this field always contains a value of 0x02 .

arg1_len

This 8-bit field contains an integer that specifies the length, in octets, of the first argument.

Subsequent fields contain the length of each sequential argument.

user

This variable length field contains the login name of the user to be authorized.

port

This variable length field contains the name of the Oracle USM port on which authorization is taking place. Following Cisco Systems convention, this field contains the string tty10 .

rem_addr

This variable length contains the location of the user to be authorized. This field contains the localhost address.

arg...

This variable length field contains a TACACS+ attribute value pair (AVP); each arg field holds a single AVP.

A TACACS+ AVP is an ASCII string with a maximum length of 255 octets. The string consists of the attribute name and its assigned value separated by either an equal sign (=) or by an asterisk (*). The equal sign (=) identifies a mandatory argument, one that must be understood and processed by the TACACS+ daemon; the asterisk (*) identifies an optional argument that may be disregarded by either the client or daemon.

Administrative authorization requires the use of only two TACACS+ AVPs: service and cmd .

The service AVP identifies the function to be authorized. In the case of the current implementation, the attribute value is always shell . Consequently the attribute takes the follow format:

service=shell

The cmd AVP identifies the specific ACLI command to be authorized. The command is passed in its entirety, from the administrative configuration root, **configure terminal**, through the final command argument. For example,

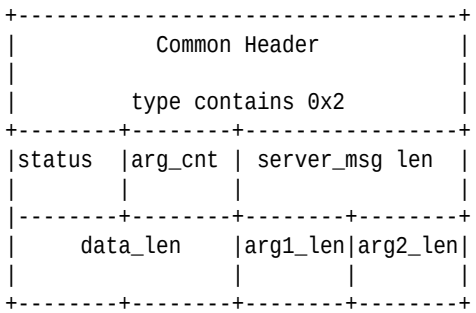
cmd=configure terminal security authentication type tacacsplus

Note the equal sign (=) used in the attribute examples, indicating that both are mandatory arguments.

Authorization RESPONSE Packet

The TACACS+ daemon sends an authorization RESPONSE packet to the Oracle USM to report authorization results.

The authorization RESPONSE packet format is as follows.



```
|   ...  |argN_len|    server_msg   |
|        |        |                 |
+--------+--------+-----------------+
|              data ...              |
+------------------------------------+
|              arg1 ...              |
+------------------------------------+
|              arg2 ...              |
+------------------------------------+
|              argN ...              |
+------------------------------------+
```

status

This 8-bit field contains an enumerated value that specifies the results of the authorization process. Supported values are 0x01 (Pass), 0x10 (Fail), and 0x11 (Error). Fail indicates that the authorization service rejected the proposed operation, while Error indicates the authorization service failed

If authorization succeeds (status=0x01), the ACLI command is executed; if authorization fails, for whatever the reason (status=0x10 or 0x11), the ACLI command is not executed, and an appropriate error message is generated.

arg_cnt

This 8-bit field contains an integer that specifies the number or arguments contained with the RESPONSE. Given the design of the current TACACS+ implementation, this field always contains a value of 0x02 .

server_msg_len

This 16-bit field contains an integer that specifies the length, in octets, of the server_msg field.

data_len

This 16-bit field contains an integer that specifies the length, in octets, of the data field.

arg1_len

This 8-bit field contains an integer that specifies the length, in octets, of the first argument.

Subsequent fields contain the length of each sequential argument.

server-msg

This optional variable length field contains a string that can be presented to the user.

data

This optional variable length field contains a string that can be presented to an administrative display, console, or log.

arg...

This optional variable length field contains a TACACS+ attribute value pair (AVP); each arg field holds a single AVP.

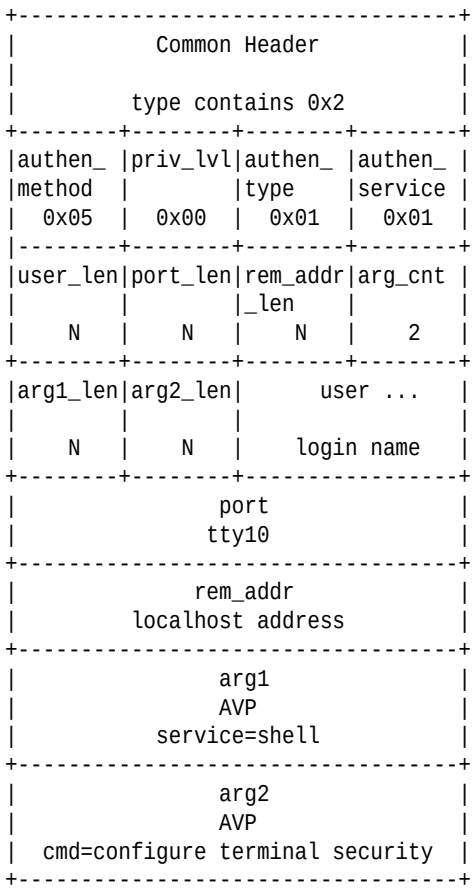
No arguments are generated in RESPONSE packets within the current TACACS+ implementation.

Authorization Scenarios

Successful and failed administrative authorization is described in terms of packet flow in the following sections.

Authorization Pass

The Oracle USM initiates the authorization with an authorization REQUEST packet.



- The authen_method field specifies the method used to authenticate the subject — 0x05 for TAC_PLUS_AUTHEN_METHOD_LOCAL (authentication by the client).
- The priv_lvl field specifies the privilege level requested by the user — 0x00 for TAC_PLUS_PRIV_LVL_MIN.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The authen_ service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The arg_cnt field contains the number of arguments in the message body.

- The arg1_len field contains the length, in octets, of the service AVP.
- The arg2_len field contains the length, in octets, of the service AVP.
- The user field contains the login name of an admin user.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The arg1 field contains the mandatory service AVP.
- The arg2 field contains the mandatory cmd AVP.

The TCACS+ daemon returns a authorization RESPONSE reporting the status, and terminating the authorization session.

```
+-----------------------------------+
|           Common Header           |
|                                   |
|         type contains 0x2         |
+--------+--------+-----------------+
| status |arg_cnt |  server_msg_len |
|  0x01  |   0    |        0        |
|--------+--------+-----------------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies the authorization status — 0x01 for TAC_PLUS_AUTHOR_STATUS_PASS_ADD (authorization approved).
- The arg_cnt field contains a value of 0 — the authorization RESPONSE returns no arguments.
- The server_msg_len and data_len fields both contain a value of 0, as required by the TACACS+ protocol.

Authorization Fail

The Oracle USM initiates the authorization with an authorization REQUEST packet.

```
+-----------------------------------+
|           Common Header           |
|                                   |
|         type contains 0x2         |
+--------+--------+--------+--------+
|authen_ |priv_lvl|authen_ |authen_ |
|method  |        |type    |service |
|  0x05  |  0x00  |  0x01  |  0x01  |
|--------+--------+--------+--------+
|user_len|port_len|rem_addr|arg_cnt |
|        |        |_len    |        |
|    N   |    N   |    N   |    2   |
+--------+--------+--------+--------+
|arg1_len|arg2_len|      user ...   |
|        |        |                 |
|    N   |    N   |    login name   |
+--------+--------+-----------------+
|               port                |
|              tty10                |
```

```
+----------------------------------+
|              rem_addr            |
|         localhost address        |
+----------------------------------+
|                arg1              |
|                AVP               |
|           service=shell          |
+----------------------------------+
|                arg2              |
|                AVP               |
|   cmd=configure terminal scurity |
+----------------------------------+
```

- The authen_method field specifies the method used to authenticate the administrative subject — 0x05 for TAC_PLUS_AUTHEN_METHOD_LOCAL (authentication by the client).
- The priv_lvl field specifies the privilege level requested by the user — 0x00 for TAC_PLUS_PRIV_LVL_MIN.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The authen_ service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem-addr field.
- The arg_cnt field contains the number of arguments in the message body.
- The arg1_len field contains the length, in octets, of the service AVP.
- The arg2_len field contains the length, in octets, of the service AVP.
- The user field contains the login name of an admin user.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The arg1 field contains the mandatory service AVP.
- The arg2 field contains the mandatory cmd AVP.

The TCACS+ daemon returns an authorization RESPONSE reporting the status, and terminating the authorization session.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x2        |
+--------+--------+--------+-------+
| status |arg_cnt |  server_msg_len |
|  0x10  |   0    |        0        |
|--------+--------+--------+--------+
|     data_len    |
|        0        |
+-----------------+
```

- The status field specifies the authorization status — 0x10 for TAC_PLUS_AUTHOR_STATUS_FAIL (authorization rejected).
- The arg_cnt field contains a value of 0 — the authorization RESPONSE returns no arguments.
- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.

TACACS+ Accounting

The Oracle USM uses TACACS+ accounting to log administrative actions. With accounting enabled, each individual ACLI command executed by an admin user is logged by the accounting service.

Accounting Message Exchange

All TACACS+ accounting packets consist of a common header and a message body. Accounting packets are of two types: REQUEST and REPLY.

The REQUEST packet has three variant forms. The START variant initiates an accounting session; the STOP variant terminates an accounting session; the WATCHDOG variant updates the current accounting session. REQUEST packets are always sent by the Oracle USM. Upon receipt of every REQUEST, the daemon must answer with a REPLY packet.

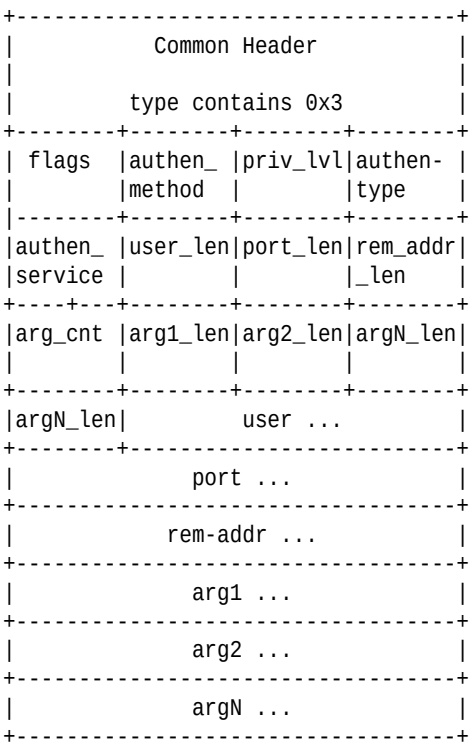
A TACACS+ accounting session proceeds as follows.

1. Immediately following successful authorization of an admin user, the Oracle USM sends an accounting REQUEST START packet.
2. The daemon responds with an accounting REPLY packet, indicating that accounting has started.
3. For each ACLI command executed by an admin user, the Oracle USM sends an accounting REQUEST WATCHDOG packet requesting accounting of the ACLI command. As the Oracle USM sends the WATCHDOG only after an admin user's access to the ACLI command is authorized, the accounting function records only those commands executed by the user, not those commands for which authorization was not granted.
4. The daemon responds with an accounting REPLY packet, indicating that the ACLI operation has been recorded by the accounting function.
5. Steps 3 and 4 are repeated for each authorized ACLI operation.
6. Immediately following logout (or timeout) of an admin user, the Oracle USM sends an accounting REQUEST STOP packet.
7. The daemon responds with an accounting REPLY packet, indicating that accounting has stopped.

Accounting REQUEST Packet

The Oracle USM, acting as a TACACS+ client, sends an accounting REQUEST START variant to the TACACS+ daemon following the successful authorization of an admin user. It sends an accounting REQUEST WATCHDOG variant to the daemon following the authorization of an admin user's access to an ACLI command. It sends an accounting REQUEST STOP variant to the daemon at the conclusion of the ACLI session.

The accounting REQUEST packet format is as follows.



flags

This 8-bit field contains an enumerated value that identifies the accounting REQUEST variant.

0x2 — START

0x4 — STOP

0x8 — WATCHDOG

authen_method

This 8-bit field contains an enumerated value that identifies the method used to authenticate the accounting subject — that is, an admin user. Because an admin user is authenticated locally by the Oracle USM, this field always contains a value of 0x05 , indicating authentication by the requesting client.

priv_lvl

This 8-bit field contains an enumerated value that identifies the privilege level associated with the accounting subject. For the current TACACS+ accounting implementation, this field always contains a value of 0x00 .

authen-type

This 8-bit field contains an enumerated value that identifies the methodology. used to authenticate the accounting subject. Because an admin user is authenticated with a simple username/password exchange, this field always contains a value of 0x01 , indicating ascii login.

authen_service

This 8-bit field contains an enumerated value that identifies the service that requested authentication. Because an admin user is authenticated with a simple username/password exchange, this field always contains a value of 0x01 , the login service.

user_len

This 8-bit field contains an integer that specifies the length, in octets, of the user field.

port_len

This 8-bit field contains an integer that specifies the length, in octets, of the port field.

rem_addr_len

This 8-bit field contains an integer that specifies the length, in octets, of the rem_addr field.

arg_cnt

This 8-bit field contains an integer that specifies the number or arguments contained with the accounting REQUEST.

arg1_len

This 8-bit field contains an integer that specifies the length, in octets, of the first argument.

Subsequent fields contain the length of each sequential argument.

user

This variable length field contains the login name of the accounting subject.

port

This variable length field contains the name of the Oracle USM port on accounting is taking place. Following Cisco System convention, this field always contains the string tty10 .

rem_addr

This variable length contains the location of the authorization subject. This field always contains the localhost address.

arg...

This variable length field contains a TACACS+ attribute value pair (AVP); each arg field holds a single AVP.

A TACACS+ AVP is an ASCII string with a maximum length of 255 octets. The string consists of the attribute name and its assigned value separated by either an equal sign (=) or by an asterisk (*). The equal sign (=) identifies a mandatory argument, one that must be understood and processed by the TACACS+ daemon; the asterisk (*) identifies an optional argument that may be disregarded by either the client or daemon.

Administrative accounting requires the use of five TACACS+ AVPs: service, task-id, start_time, and stop_time.

The task_id AVP, included in accounting REQUEST START, STOP, and WATCHDOG variants, correlates session initiation, watchdog updates, and termination packets; each associated START, STOP, and WATCHDOG packet must contain matching task-id AVPs.

task_id=13578642

The start_time AVP, included in accounting REQUEST START and WATCHDOG variants, specifies the time at which a specific accounting request was initiated. The start time is expressed as the number of seconds elapsed since January 1, 1970 00:00:00 UTC.

start_time=1286790650

The stop_time AVP, included in accounting REQUEST STOP variants, specifies the time at which a specific accounting session was terminated. The stop time is expressed as the number of seconds elapsed since January 1, 1970 00:00:00 UTC.

stop_time=1286794250

The service AVP, included in accounting REQUEST START, STOP, and WATCHDOG variants, identifies the function subject to accounting. In the case of the current implementation, the attribute value is always shell . Consequently the attribute takes the follow format:

service=shell

The cmd AVP, included in accounting REQUEST WATCHDOG variants, identifies the specific ACLI command to be processed by the accounting service. The command is passed in its entirety, from the administrative configuration root, **configure terminal**, through the final command argument. For example,

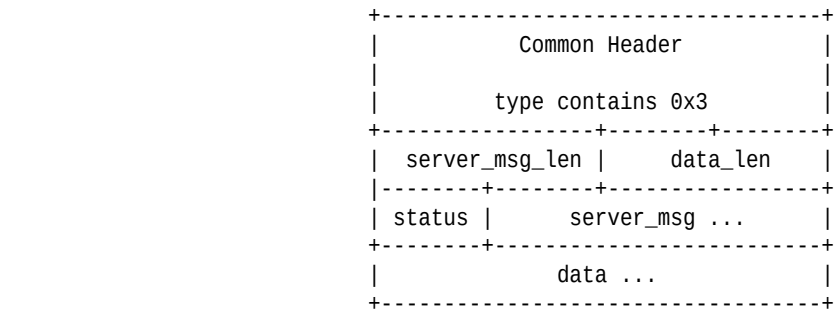
cmd=configure terminal security authentication type tacacsplus

Note the equal sign (=) used in the attribute examples, indicating that all are mandatory arguments.

Accounting REPLY Packet

The TACACS+ daemon sends an accounting REPLY packet to the Oracle USM to report accounting results.

The accounting REPLY packet format is as follows.



server_msg_len

This 16-bit field contains the length, in octets, of the server_msg field.

data_len

This 16-bit field contains the length, in octets, of the data field.

status

This 8-bit field contains the status of the previous accounting request. Supported values are:

0x1 — Success

0x2 — Error/Failure

server_msg

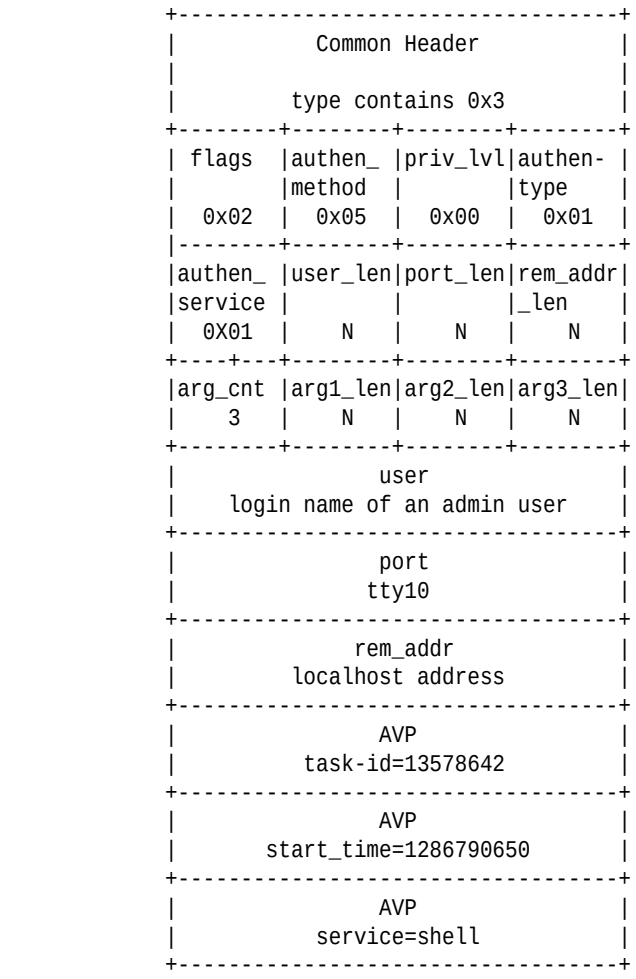
This optional variable length field can contain a message intended for display to the user. This field is unused in the current TACACS+ implementation.

data

This optional variable length field can contain miscellaneous data. This field is unused in the current TACACS+ implementation.

Accounting Scenario

The Oracle USM initiates the accounting session with an accounting REQUEST START.



- The flags field contains an enumerated value (0x02) that identifies an accounting REQUEST START.
- The authen_method field specifies the method used to authenticate the ACCOUNTING subject — 0x05 for TAC_PLUS_AUTHEN_METHOD_LOCAL (authentication by the client).
- The priv_lvl field specifies the privilege level requested by the user — 0x00 for TAC_PLUS_PRIV_LVL_MIN.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The authen_ service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.

- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The arg_cnt field contains the number of arguments in the message body.
- The arg1_len field contains the length, in octets, of the task_id AVP.
- The arg2_len field contains the length, in octets, of the start_time AVP.
- The arg3_len field contains the length, in octets, of the service AVP.
- The user field contains the login name of an admin user.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The arg1 field contains the mandatory task_id AVP.
- The arg2 field contains the mandatory start_time AVP.
- The arg3 field contains the mandatory service AVP.

The TCACS+ daemon returns an accounting REPLY reporting the status, indicating that accounting has started.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x3        |
+-----------------+----------------+
|  server_msg_len |     data_len   |
|        0        |        0       |
|--------+--------+----------------+
| status |
|  0x01  |
+--------+
```

- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.
- The status field specifies the authorization status — 0x01 for TAC_PLUS_ACCT_STATUS_SUCCESS (accounting processed).

The Oracle USM reports ACLI command execution with an accounting REQUEST WATCHDOG.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x3        |
+--------+--------+--------+--------+
| flags  |authen_ |priv_lvl|authen- |
|        |method  |        |type    |
|  0x08  |  0x05  |  0x00  |  0x01  |
|--------+--------+--------+--------+
|authen_ |user_len|port_len|rem_addr|
|service |        |        |_len    |
|  0X01  |    N   |    N   |    N   |
+----+---+--------+--------+--------+
|arg_cnt |arg1_len|arg2_len|arg3_len|
|    4   |    N   |    N   |    N   |
```

```
+--------+--------+--------+--------+
|arg4_len|           user           |
|        | login name of admin user |
+--------+--------------------------+
|                port                |
|               tty10                |
+-----------------------------------+
|              rem_addr              |
|         localhost address          |
+-----------------------------------+
|                AVP                 |
|          task-id=13578642          |
+-----------------------------------+
|                AVP                 |
|       start_time=1286790650        |
+-----------------------------------+
|                AVP                 |
|           service=shell            |
+-----------------------------------+
|                AVP                 |
|  cmd=configure terminal security   |
+-----------------------------------+
```

- The flags field contains an enumerated value (0x08) that identifies an accounting REQUEST WATCHDOG.
- The authen_method field specifies the method used to authenticate the ACCOUNTING subject — 0x05 for TAC_PLUS_AUTHEN_METHOD_LOCAL (authentication by the client).
- The priv_lvl field specifies the privilege level requested by the user — 0x00 for TAC_PLUS_PRIV_LVL_MIN.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The authen_ service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The arg_cnt field contains the number of arguments in the message body.
- The arg1_len field contains the length, in octets, of the task_id AVP.
- The arg2_len field contains the length, in octets, of the start_time AVP.
- The arg3_len field contains the length, in octets, of the service AVP.
- The arg4_len field contains the length, in octets, of the cmd AVP.
- The user field contains the login name of an admin user.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The arg1 field contains the mandatory task_id AVP.
- The arg2 field contains the mandatory start_time AVP.

- The arg3 field contains the mandatory service AVP.
- The arg4 field contains the mandatory cmd AVP.

The TCACS+ daemon returns an accounting REPLY reporting the status, indicating that the ACLI operation has been processed.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x3        |
+----------------+-----------------+
|  server_msg_len |     data_len    |
|        0        |        0        |
|--------+--------+-----------------+
| status |
|  0x01  |
+--------+
```

- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.
- The status field specifies the authorization status — 0x01 for TAC_PLUS_ACCT_STATUS_SUCCESS (accounting processed).

The Oracle USM reports an admin user logout or timeout with an accounting REQUEST STOP.

```
+----------------------------------+
|           Common Header          |
|                                  |
|         type contains 0x3        |
+--------+--------+--------+--------+
| flags  |authen_ |priv_lvl|authen- |
|        |method  |        |type    |
|  0x04  |  0x05  |  0x00  |  0x01  |
|--------+--------+--------+--------+
|authen_ |user_len|port_len|rem_addr|
|service |        |        |_len    |
|  0X01  |    N   |    N   |    N   |
+----+---+--------+--------+--------+
|arg_cnt |arg1_len|arg2_len|arg3_len|
|    3   |    N   |    N   |    N   |
+--------+--------+--------+--------+
|               user               |
|    login name of an admin user   |
+----------------------------------+
|               port               |
|              tty10               |
+----------------------------------+
|             rem_addr             |
|         localhost address        |
+----------------------------------+
|               AVP                |
|          task-id=13578642        |
+----------------------------------+
|               AVP                |
|        stop_time=1286790650      |
+----------------------------------+
|               AVP                |
|          service=shell           |
+----------------------------------+
```

- The flags field contains an enumerated value (0x04) that identifies an accounting REQUEST STOP.
- The authen_method field specifies the method used to authenticate the ACCOUNTING subject — 0x05 for TAC_PLUS_AUTHEN_METHOD_LOCAL (authentication by the client).
- The priv_lvl field specifies the privilege level requested by the user — 0x00 for TAC_PLUS_PRIV_LVL_MIN.
- The authen_type field specifies the authentication methodology — 0x01 for TAC_PLUS_AUTHEN_TYPE_ASCII (simple login).
- The authen_ service field specifies the requesting service — 0x01 for TAC_PLUS_AUTHEN_SVC_LOGIN (login service).
- The user_len field contains the length, in octets, of the user field.
- The port_len field contains the length, in octets, of the port field.
- The rem_addr_len field contains the length, in octets, of the rem_addr field.
- The arg_cnt field contains the number of arguments in the message body.
- The arg1_len field contains the length, in octets, of the task_id AVP.
- The arg2_len field contains the length, in octets, of the start_time AVP.
- The arg3_len field contains the length, in octets, of the service AVP.
- The user field contains the login name of an admin user.
- The port field contains the name of the Oracle USM port on which authentication is taking place. Following Cisco Systems convention, this field contains the string tty10 .
- The rem_addr field specifies the location of the user to be authenticated. This field contains the localhost address.
- The arg1 field contains the mandatory task_id AVP.
- The arg2 field contains the mandatory start_time AVP.
- The arg3 field contains the mandatory service AVP.

The TCACS+ daemon returns an accounting REPLY reporting the status, indicating that accounting has terminated.

```
+-----------------------------------+
|           Common Header           |
|                                   |
|         type contains 0x3         |
+-----------------+-----------------+
|  server_msg_len |     data_len    |
|        0        |        0        |
|--------+--------+-----------------+
| status |
|  0x01  |
+--------+
```

- The server_msg_len and data_len fields both contain a value of 0 , as required by the TACACS+ protocol.
- The status field specifies the authorization status — 0x01 for TAC_PLUS_ACCT_STATUS_SUCCESS (accounting processed).

TACACS+ Configuration

Configuration of TACACS+ consists of the following steps.

1. Enable TACACS+ client services
2. Specify one or more TACACS+ servers (daemons)

Enable TACACS+ Client Services

Use the following procedure to enable specific TACACS+ client AAA services.

1. Access the **authentication** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# authentication
ORACLE(authentication)#
```

2. **type** — Configure this parameter to specify the authentication protocol. The default value is **local**. Specify **tacacs** to enable the TACACS+ AAA protocol.
   - **diameter** — DIAMETER authentication (not yet supported)
   - **local** — authentication determinations are referred to a local database (default)
   - **radius** — RADIUS authentication
   - **tacacs** — TACACS+ authentication
3. **tacacs-authorization** — Configure this parameter to enable or disable command-based user authorization. The default value is **enabled** when the value of **type** is **tacacs**.
   - **disabled**
   - **enabled** (default)
4. **tacacs-authorization-arg-mode** — Configure this parameter to enable or disable sending TACACS+ authorization commands and their arguments separately to the TACACS+ server. The default value is **disabled**.
   - **disabled** (default)
   - **enabled**
5. **tacacs-accounting** — Configure this parameter to enable or disable accounting of admin ACLI operations. The default value is **enabled** when the value of **type** is **tacacs**.
   - **disabled**
   - **enabled** (default)
6. **server-assigned-privilege** — Configure this parameter to enable or disable a proprietary TACACS+ variant that, after successful user authentication, adds an additional TACACS+ request/reply exchange. During the exchange, the Security Gateway requests the privilege level of the newly authenticated user. In response, the TACACS+ daemon returns the assigned privilege level, either user or admin. Set this attribute to **enabled** to initiate the proprietary variant behavior. User accounts are denied access to the **enabled** command, thus barring them from configuration level commands. The default value is **disabled** (no privilege level information is exchanged).
   - **disabled** (default)
   - **enabled**

7. **management-strategy** — Configure this parameter to identify the selection algorithm used to choose among multiple available TACACS+ daemons. Retain the default value of **hunt** when only a single daemon is available.
   - **hunt** (default) — for the first transaction the Security Gateway selects the initially configured TACACS+ daemon. When that daemon is online and operational, the Security Gateway directs all AAA transactions to it. Otherwise, the Security Gateway selects the second-configured daemon. If the first and second daemons are offline or non-operational, the next-configured daemon is selected, and so on through the group of available daemons.
   - **roundrobin** — for the first transaction the Security Gateway selects the initially configured TACACS+ daemon. After completing the first transaction, it selects each daemon in order of configuration — in theory, evenly distributing AAA transactions to each daemon over time.
8. Type **done** to save your configuration.

Specify TACACS+ Servers

Use the following procedure to specify one or more TACACS+ servers (daemons).

1. Access the **tacacs-servers**configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# authentication
ORACLE(authentication)# tacacs-servers
ORACLE(tacacs-servers)#
```

2. Use the **address** attribute to specify the IP address of this TACACS+ daemon.

```
ORACLE(tacacs-servers)# address 172.30.0.6
ORACLE(tacacs-servers)#
```

3. Use the **port** attribute to identify the daemon port that receives TACACS+ client requests.

   Provide a port number within the range 1025 through 65535, or retain the default value, 49, the well-known TACACS+ port.

```
ORACLE(tacacs-servers)# port 49
ORACLE(tacacs-servers)#
```

4. Use the **state** attribute to specify the availability of this TACACS+ daemon.

   Select enabled (the default) or disabled.

   Only TACACS+ daemons that are in the enabled state are considered when running the server-selection algorithm.

```
ORACLE(tacacs-servers)# state enabled
ORACLE(tacacs-servers)#
```

5. Use the **realm-id** attribute to identify the realm that provides access to this TACACS+ deamon.

```
ORACLE(tacacs-servers)# realm-id accounting
ORACLE(tacacs-servers)#
```

6. Retain the default value for the **authentication-methods** attribute to specify support for all TACACS+ authentication methods (pap, chap, and ascii).
   - ascii — simple login, the Session Director prompts user for username and password

- pap — similar to ascii method, but username and password are encapsulated in a PAP header
- chap — authentication based on a shared-secret, which is not passed during the authentication process

```
ORACLE(tacacs-servers)# authentication-methods all
ORACLE(tacacs-servers)#
```

7. Use the **secret** attribute to provide the shared-secret used by the TACACS+ client and the daemon to encrypt and decrypt TACACS+ messages. The identical shared-secret must be configured on associated TACACS+ clients and daemons.

   Enter a 16-digit string, and ensure that the identical value is configured on the TACACS+ daemon.

   ```
   ORACLE(tacacs-servers)# secret 1982100754609236
   ORACLE(tacacs-servers)#
   ```

8. Use the **dead-time** attribute to specify, in seconds, the quarantine period imposed upon TACACS+ daemons that become unreachable. Quarrantined servers are not eligible to participate in the server-selection algorithm.

   Supported values are integers within the range 10 through 10000 seconds, with a default value of 10 .

   ```
   ORACLE(tacacs-servers)# dead-interval 120
   ORACLE(tacacs-servers)#
   ```

9. Type **done** to save your configuration.
10. Repeat Steps 1 through 10 to configure additional TACACS+ daemons.

> **Note:**
>
> After configuring TACACS+ daemons, complete TACACS+ configuration by compiling a list of available deamons.

11. From superuser mode, use the following command sequence to access authentication configuration mode.

    ```
    ORACLE# configure terminal
    ORACLE(configure)# security
    ORACLE(security)# authentication
    ORACLE(authentication)#
    ```

12. Use the **management-servers** attribute to identify one or more TACACS+ daemons available to provide AAA services.

    Daemons are identified by IP address and must have been previously configured as described above.

    The following example identifies three available TACACS+ daemons. Note that the list is delimited by left and right parentheses, and list items are separated by space characters.

    ```
    ORACLE(tacacs-servers)# management-servers (172.30.0.6 172.30.1.8
    172.30.2.10)
    ORACLE(tacacs-servers)#
    ```

    The following example deletes the current list.

    ```
    ORACLE(tacacs-servers)# management-servers ()
    ORACLE(tacacs-servers)#
    ```

Managing TACACS+ Operations

TACACS+ management is supported by the following utilities.

TACACS+ MIB

An Oracle proprietary MIB provides external access to TACACS+ statistics.

MIB counters are contained in the apSecurityTacacsPlusStatsTable that is defined as follows.

```
SEQUENCE {
    apSecurityTacacsPlusCliCommands                Counter32
    apSecurityTacacsPlusSuccess Authentications    Counter32
    apSecurityTacacsPlusFailureAuthentications     Counter32
    apSecurityTacacsPlusSuccess Authorizations     Counter32
    apSecurityTacacsPlusFailureAuthorizations      Counter32
}
```

apSecuritysTacacsPlusStats Table (1.3.6.1.4.1.9148.3.9.9.4)

Object Name	Object OID	Description
apSecurityTacacsCliCommands	1.3.6.1.4.1.9148.3.9.1.4.3	Global counter for ACLI commands sent to TACACS+ Accounting
apSecurityTacacsSuccess Authentications	1.3.6.1.4.1.9148.3.9.1.4.4	Global counter for the number of successful TACACS+ authentications
apSecurityTacacsFailureAuthentications	1.3.6.1.4.1.9148.3.9.1.4.5	Global counter for the number of unsuccessful TACACS+ authentications
apSecurityTacacsSuccess Authorizations	1.3.6.1.4.1.9148.3.9.1.4.6	Global counter for the number of successful TACACS+ authorizations
apSecurityTacacsFailure Authorizations	1.3.6.1.4.1.9148.3.9.1.4.7	Global counter for the number of unsuccessful TACACS+ authorizations

SNMP Trap

SNMP traps are issued when

- a TACACS+ daemon becomes unreachable
- an unreachable TACACS+ daemon becomes reachable
- an authentication error occurs
- an authorization error occurs

TACACS+ Faults

The Oracle USM supports two TACACS+ traps, apSysMgmtTacacsDownTrap and apSysMgmtTacacsDownClearTrap.

The apSysMgmtTacacsDownTrap is generated when a TACACS+ server becomes unreachable.

The apSysMgmtTacacsDownClearTrap is generated when a TACACS+ server that was unreachable becomes reachable.

The USM searches for a TACACS+ server until it finds an available one and then stops searching. However, in the TACACS+ SNMP implementation, SNMP expects the USM to make connection attempts to all servers. When there is only one TACACS+ server and that server goes down, the USM behaves normally, sending a apSysMgmtTacacsDownTrap trap when the server goes down, and a apSysMgmtTacacsDownClearTrap trap when the server comes back up. When there is more than one TACACS+ server and the active server goes down, an apSysMgmtTacacsDownTrap trap is sent, indicating that some servers are down and the next server is tried. If all servers fail, an apSysMgmtTacacsDownTrap is sent indicating that all servers are down. If one of the servers comes back up while the rest are still down, an apSysMgmtTacacsDownTrap is sent indicating that some servers are still down.

ACLI show Command

The **show tacacs stats** command displays the following statistics.

- number of ACLI commands sent for TAGACS+ accounting
- number of successful TACACS+ authentications
- number of failed TACACS+ authentications
- number of successful TACACS+ authorizations
- number of failed TACACS+ authentications
- the IP address of the TACACS+ daemon used for the last transaction

TACACS+ Logging

All messages between the Oracle USM and the TACACS+ daemon are logged in a cleartext format, allowing an admin user to view all data exchange, except for password information.

Customizing Your ACLI Settings

This section describes several ways you can customize the way you log into the ACLI and the way the ACLI displays information. Where applicable, these descriptions also contain instructions for configuration.

Disabling the Second Login Prompt

With this feature enabled, the Oracle logs you in as a Superuser (i.e., in administrative mode) regardless of your configured privilege level for either a Telnet or an SSH session. However, if you log via SSH, you still need to enter the password for local or RADIUS authentication.

Disabling the Second Login Prompt Configuration

You disable the second login prompt in the authentication configuration.

To disable the second login prompt:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **security** and press Enter.

```
ORACLE(configure)# security
ORACLE(security)#
```

3. Type **authentication** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(security)# authentication
ORACLE(authentication)#
```

4. **login-as-admin**—Set this parameter to **enabled** if you want users to be logged automatically in Superuser (administrative) mode. The default for this parameter is **disabled**.
5. Save and activate your configuration.

Persistent ACLI more Parameter

To make using the ACLI easier, the Oracle USM provides a paging feature controlled through the ACLI **cli more** command (which you can set to enabled or disabled). Disabled by default, this feature allows you to control how the Oracle USM displays information on your screen during a console, Telnet, or SSH session. This command sets the paging feature on a per session basis.

Customers who want to set the paging feature so that settings persist across sessions with the Oracle USM can set a configuration parameter that controls the paging feature. Enabling this parameter lets you set your preferences once rather than having to reset them each time you initiate a new session with the Oracle USM.

Persistent ACLI more Parameter Configuration

To set the persistent behavior of the ACLI more feature across sessions:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **system** and press Enter.

```
ORACLE(configure)# system
ORACLE(system)#
```

3. Type **system-config** and press Enter.

```
ORACLE(system)# system-config
ORACLE(system-config)#
```

If you are adding this feature to an existing configuration, you need to select the configuration (using the ACLI **select** command) before making your changes.

4. **cli-more**—Set this parameter to **enabled** if you want the ACLI more paging feature to work persistently across console, Telnet, or SSH sessions with the Oracle USM. If you want to continue to set this feature on a per session basis, leave this parameter set to **disabled** (default).
5. Save and activate your configuration.

Customized Login Banner

A text file can be put on the Oracle USM to be used as a banner to be printed before each login. The file must be called /code/banners/banner.txt. The contents of this file will be printed before each User Access Verification login sequence. The limits are that no more than 79 characters per line and no more than 20 lines from the banner.txt file will be printed.

The banner.txt file used for the ACLI customized login banner has to be saved in the /code/banners directory. If that directory does not already exist on your system, you do not have to create the directory prior to placing the banner file because the Oracle USM will create it upon boot if it does not exist.

3

System Configuration

This chapter explains how to configure system-level functionality for the Oracle USM. Both physical and network interfaces as well as general system parameters are required to configure your Oracle USM for service. Accounting functionality, SNMP configurations, trap configurations, and host routes are optional.

The following configurations are explained in this chapter:

- General system parameters—used for operating and identification purposes. In general, the informational fields have no specific effect on services, but are important to keep populated. The default gateway parameter is included here. It requires special attention since its configuration is dependent on the type of traffic the Oracle USM is servicing.
- Physical and network interfaces—enables the Oracle USM to communicate with any network element. Interfaces are one of the most basic configurations you need to create.
- SIP Interfaces and Ports—creates an interface on the Oracle USM through which the system can send, receive and process SIP messages.
- SNMP—used for monitoring system health throughout a network.
- Syslogs and Process logs—used to save a list of system events to a remote server for analysis and auditing purposes.
- Host routes—used to instruct the Oracle USM host how to reach a given network that is not directly connected to a local network interface.

General System Information

This section explains the parameters that encompass the general system information on a Oracle USM.

System Identification

Global system identification is used primarily by the Oracle USM to identify itself to other systems and for general identification purposes.

Connection Timeouts

It is important to set administrative session timeouts on the Oracle USM for security purposes. If you leave an active configuration session unattended, reconfiguration access is left open to anyone. By setting a connection timeout, only a short amount of time needs to elapse before the password is required for Oracle USM access.

Timeouts determine the specified time period that must pass before an administrative connection is terminated. Any subsequent configuration activity can only be performed after logging in again to the Oracle USM. The timeout parameter can be individually specified for Telnet sessions and for console port sessions.

After the Telnet timeout passes, the Telnet session is disconnected. You must use your Telnet program to log in to the Oracle USM once again to perform any further configuration activity.

After the console timeout passes, the console session is disconnected. The current session ends and you are returned to the login prompt on the console connection into the Oracle USM.

Configuring General System Information

This section explains how to configure the general system parameters, timeouts, and the default gateway necessary to configure your Oracle USM.

To configure general system information:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **system-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

   The following is an example what a general system information configuration might look like. Parameters not described in this section are omitted below.

   ```
   ORACLE(system-config)# show
   system-config
           hostname                      test1
           description                   Example SD
           location                      Row 3, Rack 4, Slot 451
           default-gateway               10.0.2.1
           telnet-timeout                1000
           console-timeout               1000
           last-modified-date            2004-12-08 20:15:43
   ```

   When showing a single-instance configuration element such as **system-config**, you must first use the **select** command to select the configuration element prior to viewing.

System Identification

You must specify identification parameters for this Oracle USM.

Set the following parameters to configure the system identification:

1. **hostname**—Set the primary hostname used to identify the system. This parameter is used by the software for informational purposes.
2. **description**—Enter a textual description of thesystem. This parameter is used for informational purposes.
3. **location**—Set a location description field for your system. This parameter is used for informational purposes. For example, you could include the site name and address of the location where the Oracle USM system chassis is located.
4. **default-gateway**—Set the default gateway for this Oracle USM. This is the egress gateway for traffic without an explicit destination. The application of your Oracle USM determines the configuration of this parameter.

Configuring Connection and Debug Logging Timeouts

Configure the timeouts for terminal sessions on this Oracle USM. These parameters are optional.

Set the following parameters to configure the connection timeouts:

1. **telnet-timeout**—Set the Telnet timeout to the number of seconds you want the Oracle USM to wait before it disconnects a Telnet session or an SSH session. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535
2. **console-timeout**—Set the console timeout to the number of seconds you want the Oracle USM to wait before it ends the console session. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—65535
3. **debug-timeout**—Set the time in seconds you want to use for the debug timeout. This is the time allowed before the Oracle USM times out log levels for system processes set to debug using the ACLI **notify** and **debug** commands.

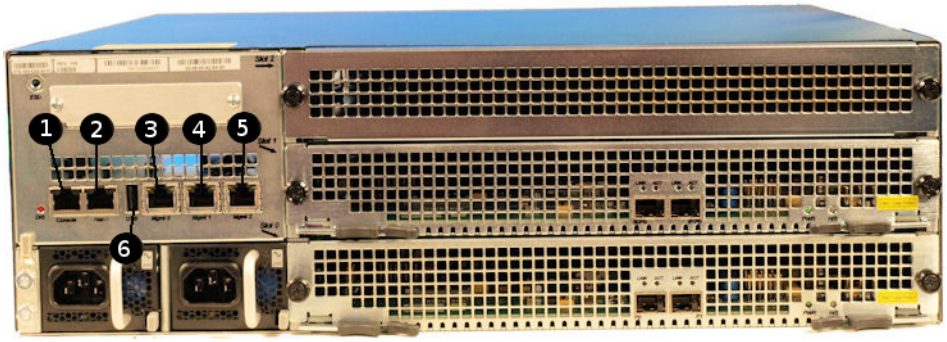
   This command does not affect log levels set in your configuration (using parameters such as **system-config>process-log-level**) or those set using the ACLI **log-level** command.

   The valid range is:
   - Minimum—0
   - Maximum—65535

Acme Packet 6300 Physical Interfaces

Acme Packet 6300 Management Interfaces

The Acme Packet 6300 includes 3 x 10/100/1000 Ethernet interfaces, a console port, and an alarm port (“dry contact alarms”). Network management interfaces are referred to as wancom0, wancom1, and wancom2. There is no front facing console port. Maintenance and control ports are located on the left-rear of the chassis, above the power supplies. See the following diagram:



1. Console port
2. Alarm port

3. wancom0
4. wancom1
5. wancom2
6. USB Port (for Acme Packet personnel access only)

Acme Packet 6300 Media Interfaces

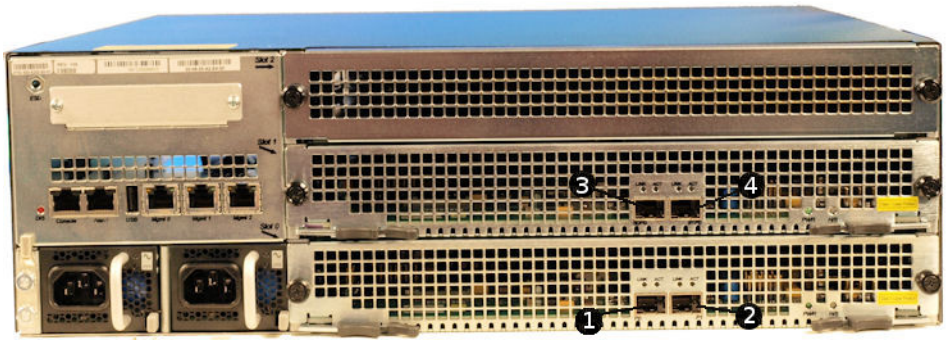
The Acme Packet 6300 has 3 PHY card slots. The bottom slot, slot0, and the middle slot, slot1, are network-facing media interfaces. The top slot, slot2, is for special processing hardware such as transcoding and is referred to as the “resource” slot. Standard Acme Packet 6300 PHY cards contain 2 x 10-gigabit Ethernet ports.

Ensure that the first 2x10Gig NIU in your system populates slot0. If you are using a second 2x10Gig NIU, it should be inserted into slot1.

> **Note:**
>
> Do not insert any 2x10Gig NIU in the resource slot (slot2).

The Acme Packet 6300 platform supports up to 4 x 10 gigabit media ports each running at full duplex line rate for all packet sizes.

Ports on the Acme Packet 6300 are numbered from left to right starting at 0. For physical interface configuration, see the following diagram



1. Slot 0, Port 0
2. Slot 0, Port 1
3. Slot 1, Port 0
4. Slot 1, Port 1

Acme Packet 4500 Physical Interfaces

There are two sets of physical interfaces on the network interface unit (NIU) used in the Acme Packet 4500. These interfaces are located on the rear of the system chassis.

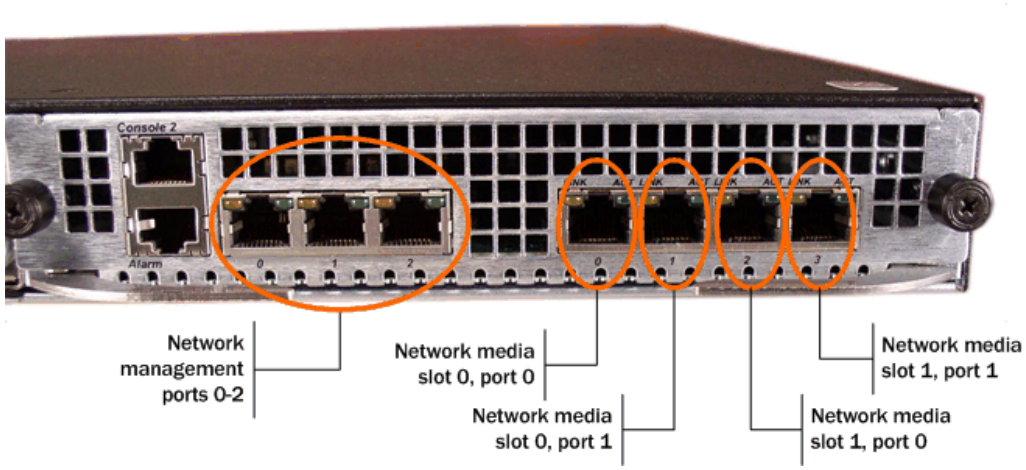
- Media interfaces are on the network interface unit (NIU); they are also referred to as network media ports
- Management interfaces are also on the NIU; they are also referred to as network management ports

The following picture of the NIU shows you how the network media and network management ports appear. These designations are an important point of reference when you set up physical interface configurations. Note that the slot parameter for network management ports will always be set to zero (0).

Network Media Interfaces

The NIU installed on your Acme Packet 4500 determines the number of interfaces, hardware protocol, and connection speed available for media and signaling traffic.

- The NIU offers either four ports , and can use single mode or multimode fiber with an LC connector.
  - 4-port GigE copper (RJ45)
  - 4-port GigE SFP (LX, SX, or Copper)
  - 4-port GigE SFP with QoS and IPSec (LX, SX, or Copper)
  - 4-port GigE SFP with IPSec (LX, SX, or Copper)
  - 4-port GigE SFP with QoS (LX, SX, or Copper)
  - 4-port GigE SFP ETC NIU (LX, SX, or Copper)



Network Management Interfaces

The first management interface (labeled port 0 on the NIU's group of management ports) is used to carry traffic such as:

- SNMP
- Telnet
- SSH
- FTP
- ACP/XML
- Logs sent from the Oracle USM
- Boot the Oracle USM from a remote file server

The other two rear interfaces (port 1 and port 2) are used for state replication for high availability (HA). For HA, these interfaces on the Acme Packet 4500 are directly connected by a crossover cable.

The following table summarizes the physical interface configuration parameters, which interface they are applicable to, and whether they are required.

Parameter	Network Media Interface	Network Management Interface
name	R	R
operation-type	R	R
port	R	R
slot	R	R
virtual-mac	O	I
admin-state	R	I
auto-negotiation	R	I
duplex-mode	R	I
speed	R	I
wancom-health-score	I	O

R = Required, O = Optional, I = Invalid

Before You Configure

This section describes steps you should take prior to configuring physical interfaces.

Before you configure a physical interface:

1. Decide on the number and type of physical interfaces you need.

   For example, you might have one media interface connecting to a private network and one connecting to the public network. You might also need to configure maintenance interfaces for HA functionality.

2. Determine the slot and port numbering you will need to enter for the physical interfaces you want to configure. The graphic above can serve as your slot and port numbering reference.

3. If you are configuring your Acme Packet 4500 for HA, refer to the HA Nodes documentation and follow the instructions there for setting special parameters in the physical interface configuration.

Physical Interface Configuration

This section describes how to configure physical interfaces.

1. Access the **phy-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)#phy-interface
   ORACLE(phy-interface)#
   ```

2. **name**—Set a name for the interface using any combination of characters entered without spaces. For example: **s0p0**.

3. **admin-state**—Leave the administrative state parameter set to **enabled** to receive and send traffic on this interface. Select **disabled** to prevent media and signaling from being received and sent. The default for this parameter is **enabled**.
4. **operation-type**—Select the type of physical interface connection to use. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface. The default value is **control**. The valid values are:
   - **media**—Use this value for configuring the media interfaces which carry production traffic.
   - **maintenance**—Use this value for configuring the management physical interfaces, used for management protocols or HA.
   - **control**—This legacy parameter may also be used to configure the management physical interfaces.
5. **slot**—Set the slot number for this physical interface. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface.
6. **port**—Set the port number for this physical interface. Refer to the appropriate platform section to identify how this parameter corresponds to an external interface.
7. **auto-negotiation**—Leave this parameter set to **enabled** so that the Oracle USM and the device to which it is linked can automatically negotiate the duplex mode and speed for the link.

   If auto-negotiation is enabled, the Oracle USM begins to negotiate the link to the connected device at the duplex mode you configure. If auto-negotiation is disabled, then the Oracle USM will not engage in a negotiation of the link and will operate only at the duplex mode and speed you set. The default is **enabled**. The valid values are:

   - enabled | disabled
8. **duplex-mode**—Set the duplex mode. The default is **full**; this field is only used if the auto-negotiation field is set to disabled.

   Given an operating speed of 100 Mbps, full duplex mode lets both devices on a link send and receive packets simultaneously using a total bandwidth of 200 Mbps. Given the same operating speed, half duplex mode limits the devices to one channel with a total bandwidth of 100 Mbps. The valid values are:

   - half | full
9. **speed**—Set the speed in Mbps of the physical interfaces; this field is only used if the auto-negotiation field is set to disabled. **100** is the default. The valid values are:
   - 10 | 100 | 1000
10. **virtual-mac**—Refer to Oracle USM High Availability (HA) documentation to learn how to set this parameter on an HA interface.
11. Type **done** to save your configuration.

Link Redundancy

Link redundancy enables the Oracle USM to run a pair of media interfaces redundantly so that in the event of a network or link failure, the Oracle USM automatically fails over to the redundant physical link. The Oracle USM polls link state on a one-second basis, so the maximum outage time is one second. And if gateway heartbeats are enabled, then gateway timeout alarms will also cause failovers.

This feature is only supported on the Acme Packet 4000 on the following NIUs:

- 4-port 10/100/1000 copper
- 4-port 1Gig SFP
- 4-port 1Gig SFP phy card with QoS

The link redundancy feature enables each slot pair (SxP1 and SxP2) on an NIU to behave as only a single port with one port as an active port and the other port as the hot standby simultaneously. Port 0 on Slots 0 and 1 is the master port, and the two Port 1s are the backup ports. The NIU receives and sends all traffic on one port, while the other acts as a standby in the event of failure. When enabled, this feature takes effect system-wide.

Link redundancy is configured by setting the **link-redundancy-state** parameter to **enabled** in **system-config**. To perform a manual switchover from one port to its redundant port, execute the **switchover-redundancy-link** command.

The criteria for port swtichover are:

- Link down event on active port
- ARP timeout to the gateway configured on the media interface
- Administratively-forced switchover

Please note the following:

- Physical interface configuration for the standby port must not exist. The network interface for the first port (port 0) should only be configured, and it becomes the preferred active port.
- A critical level ALARM will be issued during operation if both the active and standby ports experience LINK down state.
- Link redundancy is non-revertive; after switching over to the standby, if the formerly-active port recovers link, the Oracle USM does not switch back.

Link Redundancy and High Availability Interaction

The Link redundancy feature is a layer 2 feature which handles lower layer physical failure conditions automatically; the failure of one link does not cause health score decrements that result in a system-to-system switchover. However, in the event that both the active and standby ports fail on a single slot, the Oracle USMs will decrement its health score so that an active-to-standby HA switchover occurs.

The high availability (HA) feature can be considered an application layer feature which depends upon numerous critical conditions including lower layer alarm status (such as double physical link failure) to update the system's health score and determine whether to switchover. HA and LR are independent features, but they can be simultaneously configured to support extensive failover protection. You may treat them as two layers of redundancy protection. One for physical layer switchover on each slot on SBC (LR) and another (HA) as global "system" layer switchover capability.

Caveats

- Be aware that DoS protection and QoS metrics are not compatible with this feature. However, hostpath DoS protection is still available when you enable phy link redundancy.
- Link redundancy statistics are not mirrored between active and standby nodes in an HA pair.

Phy Link Redundancy Configuration

This section shows you how to enable phy link redundancy, how to force a switchover, and how to view information about the redundancy links.

Only configure port 0, the redundant port 1 is automatically configured.

1. Access the **system-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

2. Type **select** to begin editing the **system-config** object.

   ```
   ORACLE(system-config)# select
   ACMEPACKET(system-config)#
   ```

3. **link-redundancy-state**—Set this parameter to **enabled** if you want to use phy link redundancy for your system with two two-port GigE cards installed. A value of **disabled** turns this feature off. The default is **disabled**. The valid values are:

   - enabled | disabled

4. Type **done** to save your configuration.

To view link redundancy state, in Superuser mode, execute the **show redundancy link** command.

```
console# show redundancy link
Active port on Slot 0 is port: 1
Slot 0 Switchover Events: 1
Active port on Slot 1 is port: 0
Slot 1 Switchover Events: 0
```

To force a switchover, in Superuser mode, execute the **switchover-redundancy-link** and a Space and the slot number (0 or 1). This change the roles of the active and the standby ports on the slot you specify. If the command is successful, then no further information will be displayed.

```
ORACLE# switchover-redundancy-link 0
```

The system allows you to switch links only if the newly active link is up. If it is not, then the system displays information that tells you why the operation could not be completed:

```
Switch From Slot 1 Port 1, to Port 0 was not completed
Due to the fact Link State for Slot 1 Port 0 is down
```

Phy Link Redundancy (USM)

On the Acme Packet 4000 and 6000 series platforms, you can configure any NIU for phy link redundancy. Each slot pair (SxP1 and SxP2) behaves as though it has only a single port by only using one port as an active port at one time.

In this redundancy scheme, port 0 on slots 0 and 1 is the master port and port 1 is the backup port. The card receives and sends all traffic on one port, while the other acts as a standby in the event of failure. In this way, the two-port GigE card behaves as though it were a single-port card by only using one port as an active at one time.

Phy link redundancy is configured by setting the **link-redundancy-state** parameter to enabled in system-config. This feature is enabled system-wide.

You can force a switchover from one port to its redundant port using the **switchover-redundancy-link** parameter.

The value of enabling this feature is that, in the event of a network or link failure, the Oracle USM will automatically fail over to another physical link. The Oracle USM polls link state on a one-second basis, so the maximum outage you might experience prior to link failure detection and switchover is one second. And if gateway heartbeats are enabled, then gateway timeout alarms will also cause failovers.

If you are not using an HA node set-up with two Oracle USMs, then this feature can provide link-level redundancy. Even if you are using two systems as an HA node, this feature can prevent the need for one Oracle USM in the node to failover to the other by simply failing over its own link; the failure of one link does not cause health score decrements that result in a system-to-system switchover. However, in the event that both the active and standby ports fail on a single slot, the Oracle USMs will decrement its health score so that an active-to-standby switchover will occur.

Please note the following:

- Physical interface configuration for the standby port must not exist. You must configure a network interface for the first port (port 0). The configured port becomes the preferred active port.
- A critical level ALARM will be issued during operation if both the active and standby ports go LINK down.
- Link redundancy is non-revertive. Thus, after switching over to the standby, if the formerly-active port recovers link, the Oracle USM does not switch back.
- The criteria for port swtichover are:
  - Link down event on active port
  - ARP timeout to the gateway configured on the media interface
  - Administratively-forced switchover

Caveats

- Be aware that DoS protection and QoS metrics are not compatible with this feature. However, hostpath DoS protection is still available when you enable phy link redundancy.
- Link redundancy statistics are not mirrored between active and standby nodes in an HA pair.

Phy Link Redundancy Configuration

This section shows you how to enable phy link redundancy, how to force a switchover, and how to view information about the redundancy links.

Only configure port 0, the redundant port 1 is automatically configured.

1. Access the **system-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)#
```

2. Type **select** to begin editing the **system-config** object.

   ```
   ORACLE(system-config)# select
   ACMEPACKET(system-config)#
   ```

3. **link-redundancy-state**—Set this parameter to **enabled** if you want to use phy link redundancy for your system with two two-port GigE cards installed. A value of **disabled** turns this feature off. The default is **disabled**. The valid values are:

   - enabled | disabled

4. Type **done** to save your configuration.

To view link redundancy state, in Superuser mode, execute the **show redundancy link** command.

```
console# show redundancy link
Active port on Slot 0 is port: 1
Slot 0 Switchover Events: 1
Active port on Slot 1 is port: 0
Slot 1 Switchover Events: 0
```

To force a switchover, in Superuser mode, execute the **switchover-redundancy-link** and a Space and the slot number (0 or 1). This change the roles of the active and the standby ports on the slot you specify. If the command is successful, then no further information will be displayed.

```
ORACLE# switchover-redundancy-link 0
```

The system allows you to switch links only if the newly active link is up. If it is not, then the system displays information that tells you why the operation could not be completed:

```
Switch From Slot 1 Port 1, to Port 0 was not completed
Due to the fact Link State for Slot 1 Port 0 is down
```

Interface Utilization: Graceful Call Control, Monitoring, and Fault Management

When you enable this feature, the Oracle USM monitors network utilization of its media interfaces and sends alarms when configured thresholds are exceeded. You can also enable overload protection on a per-media interface basis, where the Oracle USM will prevent call initializations during high traffic but still allow established calls to continue if traffic passes the critical threshold you define.

Calculation Overview

When enabled to do so, the Oracle USM performs a network utilization calculation for each of its media ports. This calculation takes into account rates of receiving and transmitting data, the speed at which each is taking place, and the quality of data traversing the interface. The Oracle USM keeps statistics for each media port so it can compare previously- and newly-retrieved data. For heightened accuracy, calculations are performed with milliseconds (rather than with seconds).

Alarms

In the physical interface configuration, you can establish up to three alarms per media interface —one each for minor, major, and critical alarm severities. These alarms do not have an impact

on your system's health score. You set the threshold for an alarm as a percentage used for receiving and transmitting data.

For example, you might configure the following alarms:

- Minor, set to 50%
- Major, set to 70%
- Critical, Set to 90%

When the utilization percentage hits 50%, the system generates a minor alarm. At 70%, the system clears the minor alarm and issues a major one. And at 90%, the system clears the major alarm and issues a critical one. At that point, if you have overload protection enabled, the system will drop call initiations but allow in-progress calls to complete normally.

To prevent alarm thrashing, utilization must remain under the current alarm threshold for 10 seconds before the system clears the alarm and rechecks the state.

Alarm Configuration

This section shows you how to configure alarm thresholds and overload protection per media interface.

Configuring Utilization Thresholds for Media Interfaces

To configure utilization thresholds for media interfaces:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **phy-interface** and press Enter. **If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE(system)# phy-interface
   ORACLE(phy-interface)#
   ```

4. Type **network-alarm-threshold** and press Enter.

   ```
   ORACLE(phy-interface)# network-alarm-threshold
   ORACLE(network-alarm-threshold)#
   ```

5. **severity**—Enter the severity for the alarm you want to fine for this interface: **minor** (default), **major**, or **critical**. Since the parameter defaults to minor, you must change the value if you want to define a major or critical alarm.

6. **value**—Enter the percentage of utilization (transmitting and receiving) for this interface that you want to trigger the alarm. For example, you might define a minor alarm with a utilization percentage of 50. Valid values are between 0 and 100, where 0 is the default.

7. Save your work.

Configuring Graceful Call Control

You can enable the Oracle USM to stop receiving session-initiating traffic on a media interface when the traffic for the interface exceeds the critical threshold you define for it.

To enable graceful call control:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **phy-interface** and press Enter. **If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE(system)# phy-interface
   ORACLE(phy-interface)#
   ```

4. **overload-protection**—Change this parameter's value to enabled if you want to turn graceful call control on. Leave it set to disabled (default) if you do not want to use this feature.

5. Save your work.

Network Interfaces

The network interface element specifies a logical network interface. In order to use a network port on a network interface, you must configure both the physical interface and the corresponding network interface configuration elements. If the network interface does not use VLANs tagging, ensure that the **sub-port-id** parameter is set to 0, the default value. When VLAN tags are used on a network interface, the valid **sub-port-id** value can range from 1-4096. The combination of the **name** parameter and the **sub-port-id** parameter must be unique in order to identify a discrete network interface.

IP Configuration

A Oracle USM network interface has standard parameters common to nearly all IP network interfaces. There are a few fields that are unique to the Oracle USM.

VLANs

VLANs are used to logically separate a single physical interface into multiple network interfaces. There are several applications for this like MPLS VPNs (RFC 2547), MPLS LSPs, L2VPNs (IPSec, L2TP, ATM PVCs), reusing address space, segmenting traffic, and maximizing the bandwidth into a switch or router. The range of services and management capabilities you can implement with VPNs is huge.

The primary applications of VLANs on the Oracle USM are VPNs and peering. Several peering partners may terminate their connections to a Oracle USM on a single physical interface. VLAN tags are used to segregate and correctly route the terminated traffic. The Oracle USM can support a maximum of 1024 VLANs per physical interface. Ingress packets

that do not contain the correct VLAN tag will be dropped. All packets exiting on an egress interface will have the VLAN tag appended to them.

The Oracle USM can be included in an MPLS network through its connectivity to a PE router, which maps a MPLS VPN label to an 802.1q VLAN tag. Each Oracle USM can terminate different 802.1q VLANs into separate network interfaces, each of which can represent a different customer VPN.

Overlapping Networks

Overlapping networks are when two or more private networks with the same addressing schemes terminate on one physical interface. The problem this creates can easily be solved by using VLAN tagging. For example, two 10.x.x.x networks terminating on one Oracle USM network interface will obviously not work. The Oracle USM includes the IP Address, Subnet Mask, and 802.1q VLAN tag in its Network Interface determination. This allows Oracle USM to directly interface to multiple VPNs with overlapping address space.

Administrative Applications Over Media Interfaces

By default, the Oracle USM's FTP, ICMP, SNMP, SSH, and Telnet services cannot be accessed via the media interfaces. In order to enable these services, you must explicitly configure access by identifying valid source addresses for the specific applications. Doing such uses the Oracle USM's host-in-path (HIP) functionality.

When traffic is received on media interfaces, it is scanned for FTP, ICMP, SNMP, SSH, or Telnet packets. The configuration is set to identify the possible IP addresses where that traffic may be sourced from. When a match is made among packet type and source address, those packets are forwarded through the media interfaces to the processes running on the system's CPU.

Each media **network-interface**'s gateway should be configured so that off-subnet return traffic can be forwarded out the appropriate media interface. Also, it is advisable that no overlapping networks are configured between any media network interface and the administrative interfaces (wancom).

Configurable MTU Size

Configurable MTU on per network-interface basis enables the user to set a different MTU on each network interface. It also enables the user to set a system wide default MTU for IPv6 and IPv4 network interfaces. System wide defaults can be set in **system-config** configuration object by setting **ipv6-signaling-mtu** or **ipv4-signaling-mtu**. Defaults are 1500 for both IPv6 and IPv4.

These settings can be overwritten for each network interface by setting **signaling-mtu** in **network-interface** configuration object. Default is 0 – meaning use the system wide MTU.

This feature applies to all Signaling packets generated by the Oracle USM. All UDP packets greater than the MTU will be fragmented. For all TCP connections we advertise MSS (Maximum Segment Size) TCP option in accordance with the configured MTU. MSS option is sent in SYN and SYN/ACK packets to let the other side of the TCP connection know what your maximum segment size is. This ensures that no TCP packet is greater than the configured MTU.

1. MTU settings do not apply to media packets.
2. UDP: MTU settings apply only to packets sent by the Oracle USM. The Oracle USM will continue to process received packets even if they exceed to the configured MTU.

3. Security Phy (IPsec) hardware only; We subtract 100 bytes from the configured MTU to allow for extra headers added by security protocols. This happens even when Security Phy (IPsec) is in clear mode (no security is being applied). Due to hardware limitations of the Security Phy (IPsec) it only allows one MTU per physical port. The maximum MTU of all network interfaces on a given physical port will be used as the MTU for that physical port.
4. The Call Recording feature is where we make a copy of a packet, encapsulate it in an IP-in-IP header and send it to a configured Call Recording Server (CRS). When Call Recording is enabled, to allow space for IP-in-IP encapsulation we reduce the MTU of the original packets to be to be the lesser of the two options listed below.
   - Original Destination network MTU minus size of IP-in-IP header.
   - CRS network interface's MTU minus size of IP-in-IP header.

> **Note:**
>
> This will ensure that the traffic sent to the CRS will be within the MTU constraints of CRS' network-interface.

Network Interface Configuration

This section explains how to access and configure network interface.

Special Considerations

Configuration changes to network interface parameters might have an impact on boot configuration parameters. After configuring the network interface, you might receive a message indicating that you could be changing boot config parameters under the following circumstances:

- A physical interface or network interface element matches the boot interface (for example, the physical port is the same as the boot port).
- The boot configuration parameters are modified, because the IPv4 address, netmask, or gateway is different from the corresponding boot configuration parameters.

You are asked if you want to continue. If you enter yes, the configuration will be saved and then the differing boot configuration parameters will be changed. If you enter no, then the configuration is not saved and the boot configuration parameters are not changed.

Configuring the physical and network interface elements for the first management interface is optional because that interface, eth0, is implicitly created by a valid bootparam configuration that specifies the boot device, IPv4 address, subnet, and gateway.

Network Interfaces Configuration

This section describes how to configure a network interface.

1. Access the **network-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)
```

IP Configuration and Identification

You must specify the identity and address for all network interfaces.

Set the following parameters to configure a network interface:

1. **name**—Set the name for the network interface. This must be the same name as the physical interface to which it corresponds.
2. **description**—Enter a description of the network for easier identification.
3. **hostname**—Set the hostname (FQDN) of this network interface. This parameter is optional.
4. **ip-address**—Set the IP address of this network interface.
5. **netmask**—Set the netmask of this network interface in dotted decimal notation.
6. **gateway**—Set the gateway that this network interface uses to communicate with the next hop.
7. **sec-gateway**—Set an additional optional gateway for this network interface
8. **dns-ip-primary**—Set the DNS servers. You can set an additional two DNS servers by using the **dns-ip-backup1** and **dns-ip-backup2** parameters.
9. **dns-domain**—Set the default domain name.
10. **signaling-mtu**—Sets the MTU size for IPv4 or IPv6 transmission.

VLAN Configuration

One parameter is required to configure VLANs on a Oracle USM. The **sub-port-id** parameter located in the **network-interfaces** element adds and masks for a specific VLAN tag.

1. **sub-port-id**—Enter the identification of a specific virtual interface in a physical interface (e.g., a VLAN tab). If this network interface is not channelized, leave this field blank, and the value will correctly default to **0**. The **sub-port-id** is only required if the operation type is Media. The valid range is:
   - Minimum—0
   - Maximum—4095.

HIP Address Configuration

To configure administrative service functionality on a media interface, you must first define all source IP addresses in the media-interface's network that will exchange administrative traffic with the system. Next you will identify the type of administrative traffic each of those addresses will exchange.

You must configure the **gateway** parameter on this **network-interface** for administrative traffic to successfully be forwarded. You should also ensure that this network interface is not on an overlapping network as any of the administrative networks (wancoms).

Set the following parameters to configure HIP functionality on a network interface:

1. **add-hip-ip**—Configure all possible IP address(es) from which the Oracle USM will accept administrative traffic. Entries in this element are IP addresses of media network interfaces. This parameter can accept multiple IP addresses. You can later remove this entry by typing **remove-hip-ip** followed by the appropriate IP address.

2. **add-ftp-ip**—Set the IP address(es) that will access the Oracle USM's FTP server. This allows standard FTP packets enter the Oracle USM and reach the host. You can later remove this entry by typing **remove-ftp-ip** followed by the appropriate IP address.
3. **add-icmp-ip**—Set the IP address(es) that can ping the system and expect replies. This parameter can accommodate multiple ping IP addresses. You can later remove this entry by typing **remove-icmp-ip** followed by the appropriate IP address.

   For security, if the ICMP address and the hip-ip-list are not added for an address, the Oracle USM hardware discards ICMP requests or responses for the address.
4. **add-snmp-ip**—Set the IP address(es) that will access the system's SNMP process. This lets SNMP traffic enter the Oracle USM and reach the host. You can later remove this entry by typing **remove-snmp-ip** followed by the appropriate IP address.
5. **add-telnet-ip**—Set the IP address(es) that can connect and access the system through Telnet. You can later remove this entry by typing **remove-telnet-ip** followed by the appropriate IP address.
6. **add-ssh-ip**—Set the IP address(es) that can connect and access the system through SSH. You can later remove this entry by typing **remove-SSH-ip** followed by the appropriate IP address.

Configurable MTU Size

Configurable MTU on per network-interface basis enables the user to set a different MTU on each network interface. It also enables the user to set a system wide default MTU for IPv6 and IPv4 network interfaces. System wide defaults can be set in **system-config** configuration object by setting **ipv6-signaling-mtu** or **ipv4-signaling-mtu**. Defaults are 1500 for both IPv6 and IPv4.

These settings can be overwritten for each network interface by setting **signaling-mtu** in **network-interface** configuration object. Default is 0 – meaning use the system wide MTU.

This feature applies to all Signaling packets generated by the Oracle USM. All UDP packets greater than the MTU will be fragmented. For all TCP connections we advertise MSS (Maximum Segment Size) TCP option in accordance with the configured MTU. MSS option is sent in SYN and SYN/ACK packets to let the other side of the TCP connection know what your maximum segment size is. This ensures that no TCP packet is greater than the configured MTU.

1. MTU settings do not apply to media packets.
2. UDP: MTU settings apply only to packets sent by the Oracle USM. The Oracle USM will continue to process received packets even if they exceed to the configured MTU.
3. Security Phy (IPsec) hardware only; We subtract 100 bytes from the configured MTU to allow for extra headers added by security protocols. This happens even when Security Phy (IPsec) is in clear mode (no security is being applied). Due to hardware limitations of the Security Phy (IPsec) it only allows one MTU per physical port. The maximum MTU of all network interfaces on a given physical port will be used as the MTU for that physical port.
4. The Call Recording feature is where we make a copy of a packet, encapsulate it in an IP-in-IP header and send it to a configured Call Recording Server (CRS). When Call Recording is enabled, to allow space for IP-in-IP encapsulation we reduce the MTU of the original packets to be to be the lesser of the two options listed below.
   - Original Destination network MTU minus size of IP-in-IP header.
   - CRS network interface's MTU minus size of IP-in-IP header.

> **Note:**
>
> This will ensure that the traffic sent to the CRS will be within the MTU constraints of CRS' network-interface.

System Wide MTU Size

To change system wide MTU settings:

1. Access the **system-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

2. Type **select** to begin editing the **system-config** object.

   ```
   ORACLE(system-config)# select
   ACMEPACKET(system-config)#
   ```

3. **ipv6-signaling-mtu** or **ipv4-signaling-mtu** Configure MTU in the system config and optionally in the network interface. Default will be 1500 bytes.

   ```
   ORACLE(system-config)# ipv6-signaling-mtu 1500
   ORACLE(system-config)# ipv4-signaling-mtu 1600
   ```

4. Type **done** to save your configuration.

Required SIP Configuration

TheOracle USM requires that SIP processing be operational to perform basic service. Required configuration, discussed within this section, includes:

- Enable SIP Config
- Configure SIP Interfaces
- Configure SIP Ports
- Configure SIP Digest Authentication

The Oracle USM provides a myriad of other SIP controls, which you may or may not need for your deployment. See the Chapter on SIP Signaling Services for explanations of and configuration instructions to these SIP services.

Enabling SIP-Config

The Oracle USM cannot process SIP messaging if the **sip-config** is not enabled. To enable **sip-config**:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter to access the sip-config configuration elements.

```
ORACLE(session-router)# sip-config
```

4. Type **select** and press Enter to select the **sip-config** configuration element.

   ```
   ORACLE(sip-config)# select
   ```

   Selecting the **sip-config** is required, as is true of all single instance element.

5. Type **enable** and press Enter to enable the **sip-config** element.

   ```
   ORACLE(sip-config)# enable
   ```

6. Type **done** to make this setting, then save and activate your configuration.

SIP Interfaces

This section explains how to configure a SIP interface. The SIP interface defines the transport addresses (IP address and port) upon which theOracle USM receives and sends SIP messages.

Overview

The SIP interface defines the signaling interface. You can define a SIP interface for each network or realm to which the Oracle USM is connected. SIP interfaces support both UDP and TCP transport, as well as multiple SIP ports (transport addresses). The SIP interface also lets Hosted NAT Traversal (HNT) be used in any realm.

The SIP interface configuration process involves configuring the following features:

- address and transport protocols (SIP ports)
- redirect action
- proxy mode
- trust mode

About SIP Ports

A SIP port defines the transport address and protocol the Oracle USM will use for a SIP interface for the realm. A SIP interface will have one or more SIP ports to define the IP address and port upon which the Oracle USM will send and receive messages. For TCP, it defines the address and port upon which the Oracle USM will listen for inbound TCP connections for a specific realm.

You need to define at least one SIP port, on which the SIP proxy will listen for connections. If using both UDP and TCP, you must configure more than one port. For example, if a call is sent to the Oracle USM using TCP, which it needs to send out as UDP, two SIP ports are needed.

Preferred SIP Port

When a SIP interface contains multiple SIP ports of the same transport protocol, a preferred SIP port for each transport protocol is selected for outgoing requests when the specific SIP port cannot be determined. When forwarding a request that matched a cached registration entry (HNT or normal registration caching), the SIP port upon which the original REGISTER message arrived is used. Otherwise, the preferred SIP port for the selected transport protocol is used. When selecting the preferred SIP port, the default SIP port of 5060 will be selected over other non-default ports.

For SIP interfaces using the SIP NAT function, the preferred SIP port address and port will take precedence over the external address of the SIP NAT when they do not match. If both TCP and UDP SIP ports are defined, the address and port of the preferred UDP port is used.

Proxy Mode

The Oracle USM's proxy mode determines whether it forwards requests received on the SIP interface to target(s) selected from local policy; or sends a send a redirect response to the previous hop. Sending the redirect response causes the previous hop to contact the targets directly.

If the source of the request matches a session agent with a proxy mode already defined, that mode overrides the proxy mode defined in the SIP interface.

You can configure the proxy mode to use the Record-Route option. Requests for stateless and transaction operation modes are forwarded with a Record-Route header that has the Oracle USM's addresses added. As a result, all subsequent requests are routed through the Oracle USM.

Redirect Action

The redirect action is the action the SIP proxy takes when it receives a SIP Redirect (3xx) response on the SIP interface. If the target of the request is a session agent with redirect action defined, its redirect action overrides the SIP interface's.

You can set the Oracle USM to perform a global redirect action in response to Redirect messages. Or you can retain the default behavior where the Oracle USM sends SIP Redirect responses back to the previous hop (proxy back to the UAC) when the UAS is not a session agent.

The default behavior of the Oracle USM is to recurse-305-only. If the Oracle USM receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All other 3xx responses are sent back to the previous hop.

When the redirect-action parameter is set to recurse, it does so on SIP Redirect responses received from the user agent server (UAS) and sends a new request to the Contact headers contained in the SIP Redirect response.

Instead of this behavior, the Oracle USM can proxy the SIP Redirect response back to the user agent client (UAC) using the value in the session agent's redirect action field (when the UAS is a session agent). If there are too many UASes to define as individual session agents or if the UASs are HNT endpoints, and SIP Redirect responses need to be proxied for UASs that are not session agents; you can set the default behavior at the SIP Interface level.

SIP maddr Resolution

Release S-C6.2.0 provides enhanced resolution of addresses found in SIP contact headers, or in the maddr (multicast address) parameter of SIP 3xx REDIRECT messages. Previous releases resolved these addresses as either a host address or as a session agent name. With Release 6.2.0 these addresses can also be resolved as session agent group (SAG) names.

Support for SAG-based resolution is provide by a new **sip-config** parameter, **sag-lookup-on-redirect**. By default, SAG lookup is disabled, providing compatibility with prior releases.

The following sample SIP REDIRECT and ACLI configuration fragment illustrate enhanced processing.

```
Status-Line: SIP/2.0 302 Moved
Message Header
Via: SIP/2.0/UDP
192.168.200.224:5060;branch=z9hG4bKa0fs40009o90sc8oo780.1
From: <sip:1111@192.168.1.222:6000>;tag=1
To: sut <sip:2223@192.168.1.224:5060>;tag=11
Call-ID: 1-28515@192.168.1.222
CSeq: 1 INVITE
Contact: <sip:1111@192.168.1.223;maddr=test.acmepacket.com>
Privacy: user;id;critical;session
P-Preferred-Identity: sipp <sip:sipp@192.168.200.222:5060>
P-Asserted-Identity: abc.com
Subject: abc
Proxy-Require: privacy,prack,abc
Content-Length: 0

session-group
        group-name                      test.acmepacket.com
        description
        state                           enabled
        app-protocol                    SIP
        strategy                        Hunt
        dest
                                        192.168.200.222
                                        192.168.200.223
...
...
```

In this case, when the Oracle USM receives the 302, it resolves the information from maddr to a SAG name. In the above example, it will resolve to the configured SAG – test.acmepacket.com. The destinations configured in SAG test.acmepacket.com will be used to route the call.

SAG-based address resolution is based on the following set of processing rules.

1. When the Contact URI does not have an maddr parameter, and the hostname is not an IP Address, the Oracle USM will look for a SAG matching the hostname.
2. When the Contact URI has an maddr parameter that contains an IP address, the Oracle USM will not look for a SAG; it will use the IP Address as the target/next-hop.
3. When the Contact URI has an maddr parameter that contains a non-IP-address value, the Oracle USM will look for a SAG matching the maddr parameter value.

   The above logic can be turned on by enabling sag-lookup-on-redirect in the sip-config object as shown below.

SIP maddr Resolution Configuration

To configure the Oracle USM to perform SAG-based maddr resolution:

1. From superuser mode, use the following command sequence to access sip-config configuration mode. While in this mode, you configure SAG-based address resolution.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

2. Use the **sag-lookup-on-redirect** parameter to enable SAG-based maddr resolution.
3. Use **done**, **exit**, and **verify-config** to complete SAG-based address resolution.

Trust Mode

The Oracle USM supports the Calling Identity privacy requirements based on RFC 3323 and RFC 3325. The trust mode in the SIP interface determines whether the source and destination of a request is a trusted entity. With the implementation of this feature, the Oracle USM can understand and support the privacy headers and provide the capability for anonymous packets

The Oracle USM, which acts as a boundary device between the trusted platform and the untrusted Internet, understands the following headers:

- Privacy Header
- P-Asserted-Identity Header
- P-Preferred-Identity Header

Depending on the value of these headers and the mode in which the Oracle USM is being operated (B2BUA or the proxy), the appropriate actions are performed.

About the Process

On receiving a message, the Oracle USM checks whether the message source is trusted or not. It checks the SIP interface's trust mode value and, if the source is a session agent, the session agent's trust me value. Depending on these values, the Oracle USM decides whether the request's or response's source is trusted. If it receives message from a trusted source and the message contains the P-Asserted-Identity header field, the Oracle USM passes this message to the outgoing side. The outgoing side then decides what needs to be done with this request or response.

If the request or the response is received from an untrusted source, the Privacy header value is id (privacy is requested), and the P-Asserted-Identity header field is included, the Oracle USM strips the Privacy and the P-Asserted-Identity headers and passes the request or the response to the outgoing side.

If the request or the response contains the P-Preferred-Identity header and the message source is untrusted, the Oracle USM strips the P-Preferred-Identity header from the request or the response and passes the message to the outgoing side.

If the source is trusted or privacy is not requested (the value of the Privacy Header is not id) and the request or the response contains the P-Preferred-Identity header, the Oracle USM performs the following actions:

- inserts the P-Asserted-Identity header field with the value taken from the P-Preferred-Identity header field
- deletes the P-Preferred-Identity header value
- passes this request or the response to the Outgoing side for the appropriate action, depending on the whether the destination is trusted or not

After the Oracle USM passes the request or the response to the outgoing side, it checks whether the destination is trusted by checking the SIP interface's trust mode value and the session agent's trust me value (if the destination is configured as session agent).

- The destination is trusted
  The Oracle USM does nothing with the request or the response and passes it to the destination. If the P_Asserted_Identity headers are present, they are passed to the session agent (if the destination is configured as session agent).
- The destination is untrusted

The Oracle USM looks at the value of the Privacy header. If set to id, the Oracle USM removes all the P-Asserted-Identity headers (if present). It strips the Proxy-Require header if it is set to privacy. The Oracle USM also sets the From field of SIP header to Anonymous and strips the Privacy header.

If the Privacy header is set to none, the Oracle USM does not remove the P-Asserted-Identity header fields.

If there is no Privacy header field, the SD will not remove the P-Asserted-Identity headers.

To implement this feature, you need to configure the session agent's trust me parameter to enabled (if the message source is a session agent) and the SIP interface's trust mode to the appropriate value.

Configurable Timers and Counters

SIP timers and counters can be set in the global SIP configuration, and two can be specific for individual SIP interfaces.

You can set the expiration times for SIP messages, and you can set a counter that restricts the number of contacts that the Oracle USM tries when it receives a REDIRECT. These are similar to two parameters in the global SIP configuration, trans-expire and invite-expire. You can also set a parameter that defines how many contacts/routes the Oracle USM will attempt on redirect.

Timer to Tear Down Long Duration Calls

The Oracle USM currently provides the "flow-time-limit" timer to terminate long duration calls. However, this timer is reset whenever the Oracle USM receives a Re-INVITE or UPDATE message, even when it is provided for the session timer. This feature adds a non-resettable timer that, when enabled and upon expiration, tears down long duration calls.

When the "flow-time-limit" timer for terminating long media flow expires, the Oracle USM sends a SIP BYE or H323 Release Complete message to tear down the long call. However, this timer is reset whenever the SBC receives a Re-INVITE or UPDATE message. In most call scenarios, long duration calls are terminated with the expiration of this timer, but there are some cases where a call can stay connected for a longer duration. For example, if a user connects to an IVR service and does not hang up the phone receiver properly, there is no way for the network provider to free up the IVR resources if the user devices send session updating requests. To prevent this situation, this feature adds a new timer **session-max-life-limit**, which starts when the call or session is established and does not reset for any session update, keep-alive or SBC switchover. On expiry, the call is torn down if it's in established state.

The new timer **session-max-life-limit** can be provisioned in the following configuration elements, in order of precedence from highest to lowest: **session-agent**, **realm-config**, **sip-interface**, and **sip-config**. Its range of values is {0-2073600} seconds with an additional special case value of "Unlimited", which is treated as the highest possible value. The default value is 0 (no timer).

Difference between 0 and Unlimited

No timer is created when **session-max-life-limit** is configured to either the value 0 or "Unlimited", so no timeout can occur. The difference between the two values is how they are handled when determining which value of **session-max-life-limit** to use when there are several specified within the various configuration elements. When a session is created the timer examines both the ingress side and the egress side and, in cases where both sides have a configured value for **session-max-life-limit**, uses the side with the lower (stricter) value. On each side, the SBC reviews the configuration elements relevant to the session and uses the

value of **session-max-life-limit** from the configuration element with the highest precedence (**session-agent**, then **realm-config**, then **sip-interface**, and lastly**sip-config**). When the value is set to 0, the configuration element is ignored and the next configuration element in the precedence chain is looked at. A value between 1 and 2073600 (24 days) or the value "Unlimited" is treated as a valid configured value. In this case the SBC will not move onto the next element in the precedence chain and the value is used in the final comparison between the egress and ingress values. The value "Unlimited" is viewed as the highest possible value, and therefore is considered greater than any other value it is compared against. The value 0 is skipped over and completely ignored.

For example, on the ingress side the value of **session-max-life-limit** in **realm-config** is set to 86400 and the value of **session-max-life-limit** in **session-agent** is set to "Unlimited". The **session-agent** value has a higher precedence than the **realm-config** value so, therefore, the value "Unlimited" is used for the ingress side. On the egress side the value of **session-max-life-limit** in **realm-config** is set to 43200 and the value of **session-max-life-limit** in **session-agent** is set to 0 (no timer), so the value of **session-max-life-limit** in **realm-config** is used. When compared against the ingress side the value 43200 is less than "Unlimited"; therefore, the value set for the timer is 43200.

Timer to Tear Down Long Duration Calls Configuration

Use the following procedure to configure, in the **session-agent**, **realm-config**, **sip-interface**, and **sip-config** configuration elements, a non-resettable timer that, when enabled and upon expiration, tears down long duration calls.

Although the timer occurs in four separate configuration elements, for brevity only the procedure for configuring **realm-config** is shown as the general procedure does not vary for the other configuration elements.

For the **realm-config** configuration element:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **session-max-life-limit** — Enter the maximum interval in seconds before the SBC must terminate long duration calls. The value supercedes the value of **session-max-life-limit** in the **sip-interface** and **sip-config** configuration elements and is itself superceded by the value of **session-max-life-limit** in the **session-agent** configuration element. The default value is 0 (off/ignored).
4. Type **done** to save your configuration.

SIP Interface Configuration

To configure a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **state**—Enable or disable the SIP interface. The default is **enabled**. The valid values are:

   - enabled | disabled

5. **realm-id**—Enter the name of the realm to which the SIP interface is connected.

6. **sip-ports**—Access the **sip-ports** subelement.

7. **carriers**—Enter the list of carriers related to the SIP interface.

   Entries in this field can be from 1 to 24 characters in length and can consist of any alphabetical character (Aa-Zz), numerical character (0-9), or punctuation mark (! ”$ % ^ & * ( ) + - = < > ? ‘ | { } [ ] @ / \ ‘ ~ , . _ : ;) or any combination of alphabetical characters, numerical characters, or punctuation marks. For example, both 1-0288 and acme_carrier are valid carrier field formats

8. **proxy-mode**—Enter an option for the proxy mode parameter. Valid values are:

   - **proxy**—Forward all SIP requests to selected targets.
   - **redirect**—Send a SIP 3xx redirect response with the selected target(s) in the Contact header.
   - **record-route**—Forward requests to selected target(s) and insert a Record-Route header with the Oracle USM’s address. For stateless and transaction mode only.

9. **redirect-action**—Enter the value for the redirect action. Valid values are:

   - **recurse-305-only**—(default) If the Oracle USM receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All other 3xx responses are sent back to the previous hop.
   - **proxy**—Send the SIP request back to the previous hop.
   - **recurse**—Recurses on the Contacts in the response.

     The designated proxy action will apply to SIP 3xx responses received from non-session agents and to 3xx responses received from session agents without configured SIP Redirect message actions (for example, session agents without values for the redirect action field).

10. **contact-mode**—Set the Contact header routing mode, which determines how the contact address from a private network is formatted.

    For example, whether a maddr parameter equal to the Oracle USM’s SIP proxy needs to be added to a URI present in a Contact header.

    The default is **none**. The valid values are:

    - **none**—The address portion of the header becomes the public address of that private realm.

- **maddr**—The address portion of the header will be set to the IP address of the Oracle USM's B2BUA.
- **strict**—The contents of the Request-URI is destroyed when a Record-Route header is present.
- **loose**—The Record-Route header is included in a Request, which means the destination of the request is separated from the set of proxies that need to be visited along the way.

11. **nat-traversal**—Define the type of HNT enabled for SIP. The default is **none**. Valid values are:
    - **none**—HNT function is disabled for SIP.
    - **rport**—SIP HNT function only applies to endpoints that include the rport parameter in the Via header. HNT applies when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address.
    - **always**—SIP HNT applies to requests when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address. (Even when the rport parameter is not present.)
12. **nat-interval**—Set the expiration time in seconds for the Oracle USM's cached registration entry for an HNT endpoint. The default is **30**. The valid range is:
    - Minimum—0
    - Maximum—999999999

    Oracle recommends setting the NAT interval to one-third of the NAT binding lifetime. A NAT binding lifetime is the network connection inactivity timeout. The value is configured (or hardwired) in the NAT device (firewall). This timer is used to cause the UA to send REGISTER messages frequently enough to retain the port binding in the NAT. Retaining the binding lets inbound requests to be sent through the NAT.
13. **tcp-nat-interval**—Set the registration cache expiration time in seconds to use for endpoints behind a NAT device that register using TCP. On upgrade, the Oracle USM assigns this parameter the same value as the existing NAT interval. The default is **90**. The valid range is:
    - Minimum—0
    - Maximum—999999999

    The Oracle USM uses the value you set for the TCP NAT interval as the expiration value passed back in SIP REGISTER (200 OK) responses to endpoints behind a NAT that register over TCP. The NAT interval value with which you are familiar from previous releases is used for endpoints behind a NAT that register over UDP. Requiring endpoints that register over TCP to send refresh requests as frequently as those registering over UDP puts unnecessary load on the Oracle USM. By adding a separate configuration for the TCP NAT interval, the load is reduced.

    For upgrade and backward compatibility with Oracle USM releases prior to Release 4.1, when the tcpNatInterval is not present in the XML for a SIP interface configuration, the value of the NAT interval (natInterval) is used for the TCP NAT interval as well.
14. **registration-caching**—Enable for use with all UAs, not just those that are behind NATs. The default is **disabled**. The valid values are:
    - enabled | disabled

If enabled, the Oracle USM caches the Contact header in the UA's REGISTER request when it is addressed to one of the following:

- Oracle USM
- registrar domain value
- registrar host value

The Oracle USM then generates a Contact header with the Oracle USM's address as the host part of the URI and sends the REGISTER to the destination defined by the registrar host value.

Whether or not SIP HNT functionality is enabled affects the value of the user part of the URI sent in the Contact header:

- HNT enabled: the Oracle USM takes the user part of the URI in the From header of the request and appends a cookie to make the user unique. A cookie is information that the server stores on the client side of a client-server communication so that the information can be used in the future.
- HNT disabled: the user part of the Contact header is taken from the URI in the From header and no cookie is appended. This is the default behavior of the Oracle USM.

When the registrar receives a request that matches the address-of-record (the To header in the REGISTER message), it sends the matching request to the Oracle USM, which is the Contact address. Then, the Oracle USM forwards the request to the Contact-URI it cached from the original REGISTER message.

15. **min-reg-expire**—Set the time in seconds for the SIP interface. The value you enter here sets the minimum registration expiration time in seconds for HNT registration caching. The default is **300**. The valid range is:

    - Minimum—0
    - Maximum—999999999

    This value defines the minimum expiration value the Oracle USM places in each REGISTER message it sends to the real registrar. In HNT, the Oracle USM caches the registration after receiving a response from the real registrar and sets the expiration time to the NAT interval value.

    Some UAs might change the registration expiration value they use in subsequent requests to the value specified in this field. This change causes the Oracle USM to send frequent registrations on to the real registrar.

16. **registration-interval**—Set the Oracle USM's cached registration entry interval for a non-HNT endpoint. Enter the expiration time in seconds that you want the Oracle USM to use in the REGISTER response message sent back to the UA. The UA then refreshes its registration by sending another REGISTER message before that time expires. The default is **3600**. The valid range is:

    - Minimum—0

      A registration interval of zero causes the Oracle USM to pass back the expiration time set by and returned in the registration response from the registrar.

    - Maximum—999999999

      If the expiration time you set is less than the expiration time set by and returned from the real registrar, the Oracle USM responds to the refresh request directly rather than forwarding it to the registrar.

      Although the registration interval applies to non-HNT registration cache entries, and the loosely related NAT interval applies to HNT registration cache entries, you can use

the two in combination. Using a combination of the two means you can implement HNT and non-HNT architectures on the same Oracle USM. You can then define a longer interval time in the registration interval field to reduce the network traffic and load caused by excess REGISTER messages because there is no NAT binding to maintain.

17. **route-to-registrar**—Enable routing to the registrar to send all requests that match a cached registration to the destination defined for the registrar host; used when the Request-URI matches the registrar host value or the registrar domain value, not the Oracle USM's address. Because the registrar host is the real registrar, it should send the requests back to the Oracle USM with the Oracle USM's address in the Request-URI. The default is **disabled**. The valid values are:

    - enabled | disabled

      For example, you should enable routing to the registrar if your network uses a N Oracle USM and needs requests to go through its service proxy, which is defined in the registrar host field.

18. **teluri-scheme**—Enable to convert SIP URIs to tel (resources identified by telephone numbers) URIs.

    If enabled, the requests generated on this SIP interface by the Oracle USM will have a tel URI scheme instead of the SIP URI scheme. Only the Request, From, and To URIs are changed to the tel scheme. After the dialog is established, the URIs are not changed. The default is **disabled**. The valid values are:

    - enabled | disabled

19. **uri-fqdn-domain**—Change the host part of the URIs to the FQDN value set here. If set to enabled, and used with an FQDN domain/host, the requests generated by the Oracle USM on this SIP interface will have the host part of the URI set to this FQDN value. Only the Request, To, and From URIs are changed. After the dialog is established, the URIs are not changed.

20. **trust-mode**—Set the trust mode for the SIP interface, which is checked by the Oracle USM when it receives a message to determine whether the message source is trusted. The default is **all**. Available options are:

    - **all**—Trust all SIP elements (sources and destinations) in the realm(s), except untrusted session agents. Untrusted session agents are those that have the **trust-me** parameter set to **disabled**.
    - **agents-only**—Trust only trusted session agents. Trusted session agents are those that have the **trust-me** parameter set to **enabled**.
    - **realm-prefix**—Trust only trusted session agents, and source and destination IP addresses that match the IP interface's realm (or subrealm) address prefix. Only realms with non-zero address prefixes are considered.
    - **registered**—Trust only trusted session agents and registered endpoints. Registered endpoints are those with an entry in the Oracle USM's registration cache.
    - **none**—Trust nothing.

      Session agents must have one or more of the following:

    - global realm
    - same realm as the SIP interface
    - realm that is a subrealm of the SIP interface's realm

21. **trans-expire**—Set the TTL expiration timer in seconds for SIP transactions. This timer controls the following timers specified in RFC 3261:
    - Timer B—SIP INVITE transaction timeout
    - Timer F—non-INVITE transaction timeout
    - Timer H—Wait time for ACK receipt
    - Timer TEE—Used to transmit final responses before receiving an ACK

    The default is **0**. If you leave this parameter set to the default, then the Oracle USM uses the timer value from the global SIP configuration. The valid range is:
    - Minimum—0
    - Maximum—999999999
22. **invite-expire**—Set the TTL expiration timer in seconds for a SIP client/server transaction after receiving a provisional response.

    You set this timer for the client and the sever by configuring it on the SIP interface corresponding to the core or access side.

    The default is **0**. If you leave this parameter set to the default, then the Oracle USM uses the timer value from the global SIP configuration. The valid range is:
    - Minimum—0
    - Maximum—999999999
23. **max-redirect-contacts**—Set the maximum number of contacts or routes for the Oracle USM to attempt in when it receives a SIP Redirect (3xx Response). The default is **0**. If you leave this parameter set to the default, then the Oracle USMwill exercise no restrictions on the number of contacts or routes. The valid range is:
    - Minimum—0
    - Maximum—10
24. **response-map**—Enter the name of the SIP response map configuration that you want to apply to this SIP interfaces for outgoing responses. This parameter is blank by default.
25. **local-response-map**—Enter the name of the SIP response map configuration that you want to apply to this SIP interfaces for locally-generated SIP responses. This parameter is blank by default.
26. **options**—Optional.

Configuring SIP Ports

To configure SIP ports:

1. From sip-interface, type **sip-ports** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

    ```
    ORACLE(sip-interface)# sip-ports
    ORACLE(sip-port)#
    ```
2. **address**—Enter the IP address of the host associated with the sip-port entry on which to listen. For example:

    ```
    192.168.11.101
    ```
3. **port**—Enter the port number you want to use for this sip-port. The default is **5060**. The valid range is:

- Minimum—1025
- Maximum—65535

4. **transport-protocol**—Indicate the transport protocol you want to associate with the SIP port. The default is **UDP**. The valid values are:

   - **TCP**—Provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with the retransmission of packets when necessary.
   - **UDP**—Provides a simple message service for transaction-oriented services. Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts.
   - **TLS**—See the Security chapter for more information about configuring TLS.

5. **allow-anonymous**—Define the allow anonymous criteria for accepting and processing a SIP request from another SIP element.

   The anonymous connection mode criteria includes admission control based on whether an endpoint has successfully registered. Requests from an existing SIP dialog are always accepted and processed. The default is **all**.

   The following table lists the available options.

   - **all**—All requests from any SIP element are allowed.
   - **agents-only**—Only requests from configured session agents are allowed. The session agent must fit one of the following criteria:
     - Have a global realm.
     - Have the same realm as the SIP interface
     - Be a sub-realm of the SIP interface's realm.
       When an agent that is not configured on the system sends an INVITE to a SIP interface, the Oracle USM:
     - Refuses the connection in the case of TCP.
     - Responds with a 403 Forbidden in the case of UDP.
   - **realm-prefix**—The source IP address of the request must fall within the realm's address prefix or a SIP interface sub-realm. A sub-realm is a realm that falls within a realm-group tree. The sub-realm is a child (or grandchild, and so on) of the SIP interface realm.

     Only realms with non-zero address prefixes are considered. Requests from session agents (as described in the **agents-only** option) are also allowed.
   - **registered**—Only requests from user agents that have an entry in the registration cache (regular or HNT) are allowed; with the exception of a REGISTER request. A REGISTER request is allowed from any user agent.

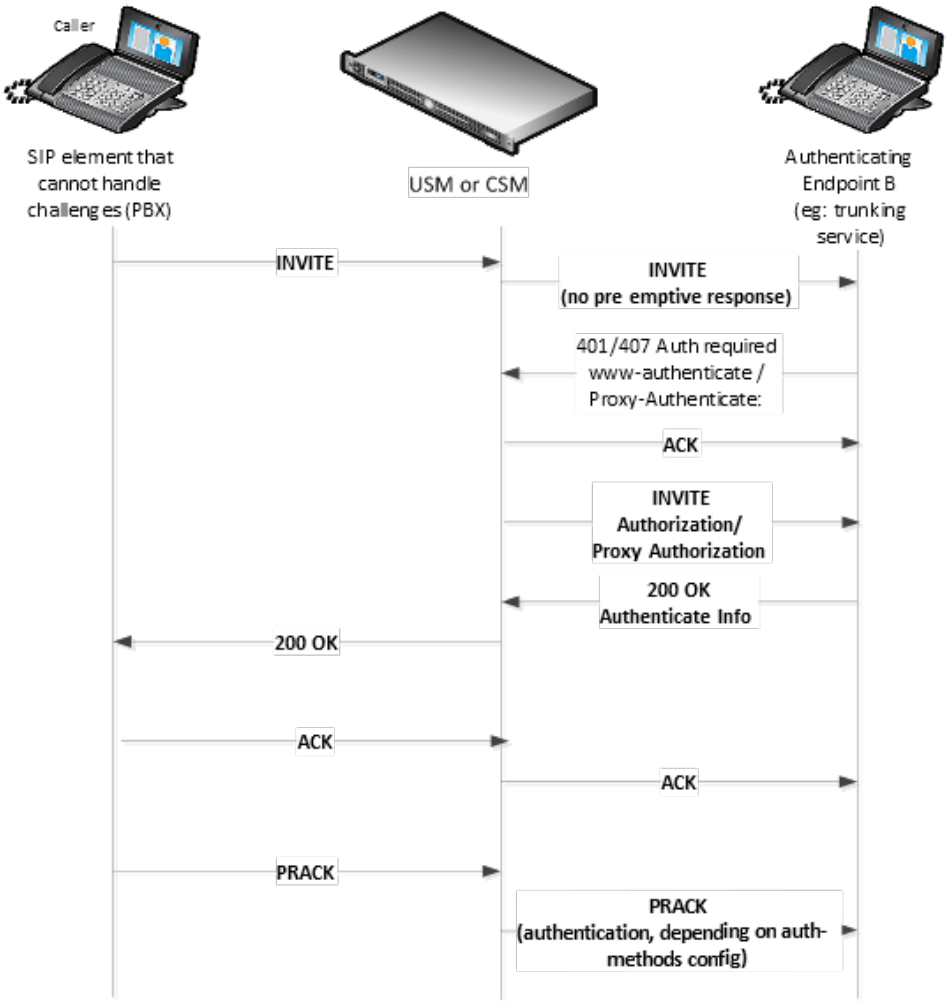
     The registration cache entry is only added if the REGISTER is successful. Requests from configured session agents (as described in the **agents-only** option) are also allowed.
   - **register-prefix**—Only requests from user agents that have an entry in the Registration Cache (regular or HNT) are allowed; with the exception of a REGISTER request. A REGISTER request is allowed only when the source IP address of the request falls within the realm address-prefix or a SIP interface sub-realm. Only realms with non-zero address prefixes are considered.

The Registration Cache entry is only added if the REGISTER is successful. Requests from configured session agents (as described in the **agents-only** option) are also allowed.

Digest Authentication with SIP

Digest authentication for Session Initiation Protocol (SIP) is a type of security feature on the Oracle USM that provides a minimum level of security for basic Transport Control Protocol (TCP) and User Datagram Protocol (UDP) connections. Digest authentication verifies that both parties on a connection (host and endpoint client) know a shared secret (a password). This verification can be done without sending the password in the clear.

Digest authentication is disabled by default on the Oracle USM. When digest authentication is enabled, the Oracle USM (host) responds to authentication challenges from SIP trunking Service Providers (endpoint client). The Oracle USM performs authentication for each IP-PBX initiating the call. However, the authentication challenge process takes place between the host and the client only since the IP-PBX cannot handle authentication challenges. The following illustration shows the digest authentication process.



The digest authentication scheme is based on a simple challenge-response paradigm. A valid response contains a checksum (by default, the MD5 checksum) of the “username” and password. In this way, the password is never sent in the clear.

By default, the Oracle USM uses cached credentials for all requests within the same dialog, once the authentication session is established with a 200OK from the authenticating SIP element. If the in-dialog-methods attribute contains a value, it specifies the requests that have challenge-responses inserted within a dialog.

In digest authentication with SIP, the following can happen:

- More than one authenticating SIP element (IP-PBX) may be the destination of requests.
- More than one authentication challenge can occur in a SIP message. This can occur when there are additional authenticating SIP elements behind the first authenticating SIP element.
- The Oracle USM distinguishes whether the IP-PBX is capable of handling the challenge. If Digest Authentication is disabled (no auth-attributes configured) on the Session Agent, the challenge is passed back to the IP-PBX.

> **Note:**
>
> If there are multiple challenges in the request, and if the Oracle USM has only some of the cached credentials configured, the Oracle USM adds challenge-responses for the requests it can handle, and does not pass the challenge back to the IP-PBX.

Challenge-Responses in Requests not in the Dialog

A digest authentication session starts from the client response to a www-authenticate/proxy-authenticate challenge and lasts until the client receives another challenge in the protection space defined by the auth-realm. Credentials are not cached across dialogs; however, if a User Agent (UA) is configured with the auth-realm of its outbound proxy, when one exists, the UA may cache credentials for that auth-realm across dialogs.

> **Note:**
>
> Existing Oracle USM behavior with surrogate-agents is that they cache credentials from REGISTER for INVITE sessions only if the Oracle USM is considered a UA sending to its outbound proxy.

Surrogate Agents and the Oracle USM

In the case where a surrogate-agent is configured for the IP-PBX, you do not have to configure digest authentication attributes in the session-agent object for the same IP-PBX. The surrogate-agent authentication configuration takes precedence over the session-agent authentication configuration and so it is ignored.

The following illustration shows an example of a surrogate-agent with a session-agent in the network.

Configuring Digest Authentication

In the Oracle USM ACLI, you can access the Digest Authentication object at the path session-router->session-agent->**auth-attribute**. If enabled, the Digest Authentication process uses the attributes and values listed in this table.

> **Note:**
>
> If enabling Digest Authentication, all attributes listed below are required except for the in-dialog-methods attribute which is optional.

The following table lists the digest authentication object

```
ORACLE(auth-attribute)# show
        auth-attribute
                auth-realm                      realm01
                username                        user
                password                        ********
                in-dialog-methods               ACK INVITE SUBSCRIBE
```

To configure digest authentication on the Oracle USM:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type session-router and press Enter to access the session router-related objects.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type session-agent and press Enter to access the session agent-related attributes.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Type auth-attribute and press Enter to access the digest authentication-related attributes.

   ```
   ORACLE(session-agent)# auth-attribute
   ORACLE(auth-attribute)#
   ```

5. auth-realm — Enter the name (realm ID) of the host realm initiating the authentication challenge. This value defines the protected space in which the digest authentication is performed. Valid value is an alpha-numeric character string. Default is blank.

   ```
   ORACLE(auth-attribute)# auth-realm realm01
   ```

6. username — Enter the username of the client. Valid value is an alpha-numeric character string. Default is blank.

   ```
   ORACLE(auth-attribute)# username user
   ```

7. password — Enter the password associated with the username of the client. This is required for all LOGIN attempts. Password displays while typing but is saved in clear-text (i.e., *****). Valid value is an alpha-numeric character string. Default is blank.

```
ORACLE(auth-attribute)# password *******
```

8. in-dialog-methods — Enter the in-dialog request method(s) that digest authentication uses from the cached credentials. Specify request methods in a list form separated by a space enclosed in parentheses. Valid values are:

- INVITE | BYE | ACK | CANCEL | OPTIONS | SUBSCRIBE | PRACK | NOTIFY | UPDATE | REFER

```
ORACLE(auth-attribute)# in-dialog-methods (ack invite subscribe)
```

> **Note:**
>
> The methods not in this list are still resubmitted if a 401/407 response is received by the Oracle USM.

If you do not specify any in-dialog-method value(s), digest authentication does not add challenge-responses to in-dialog requests within a dialog.

This attribute setting applies to in-dialog requests only.

Additional Notes

The following are additional notes that describe the digest authentication process:

- The Oracle USM always challenges the first LOGIN request, and initial authentication begins with that request. The recalculated authorization key — the credentials — are then included in every subsequent request.
- If the Oracle USM does not receive any communication from the client within the expiration period, the Oracle USM logs the client out and tears down the transport connection. Faced with interface loss, the Oracle USM default behavior is to flush all warrant information from the target database. This response necessitates that the client first login/re-register with the Oracle USM, and then repopulate the empty database using a series of ADD requests. This behavior ensures that client and Oracle USM target databases are synchronized.

  Alternatively, when faced with interface loss, the Oracle USM can retain all warrant information within the target database. This response necessitates only that the client first login/re-register with the Oracle USM. After successful registration the client should, but is not required to, use a series of GET, ADD, and DELETE requests to ensure that the Oracle USM and client target databases are synchronized.
- The Oracle USM ignores the Authentication-Info header that comes in the 200OK response after digest authentication is complete. The Oracle USM receives a 401/407 response from the client. However, some surrogate-agents may process the Authentication-Info header in a single challenge.

Digest Authentication and High Availability

The Oracle USM supports digest authentication in high availability (HA) environments. The session-agent configuration, which includes the digest authentication parameters on the primary Oracle USM, are replicated on the HA Oracle USM. However, cached credentials on the primary device are not replicated on the HA device.

IP Identification (ID) Field

By default, non-fragmented UDP packets generated by media interfaces have the ID field set to 0. You can configure the Oracle USM to populate this field with an incrementing value by adding the **increment-ip-id** option in the media manager. Every non-fragmented packet sent will have its ID increased by one from the previous packet sent.

Using a packet trace application, egress packets from the Oracle USM will have an ID field that appears to be incrementing. Enabling the ID field can help distinguish a retransmitted non-fragmented application layer packet from a packet retransmitted by the network layer in monitoring or lab situations.

IP Identification Field Configuration

To enable ID field generation in media-manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. **options**—Set the options parameter by typing **options**, a Space, the option name **increment-ip-id** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(media-manager)# options + increment-ip-id
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

4. Save and activate your configuration.

SNMP

This section explains how to configure Simple Network Management Protocol (SNMP), trap receivers, and syslog servers. These features are not essential for baseline Oracle USM service, but they are necessary to use network management systems to manage the Oracle USM. They provide important monitoring and system health information that contribute to a robust deployment of the system.

Overview

SNMP is used to support monitoring of network-attached devices for conditions that warrant administrative attention. SNMP is comprised of three groups of settings on a Oracle USM. These settings are system-wide configurations including MIB contact information, SNMP community settings, and trap receivers.

Basic SNMP Parameters

The Oracle USM includes several parameters that control basic SNMP functionality. The MIB-related elements are for informational purposes, and are helpful if set. The remainder of the parameters determines if certain Oracle USM events are reported to the SNMP system.

SNMP Community

An SNMP community is a grouping of network devices and management stations used to define where information is sent and accepted. An SNMP device or agent might belong to more than one SNMP community. SNMP communities provide a type of password protection for viewing and setting management information within a community.

SNMP communities also include access level settings. They are used to define the access rights associated with a specific SNMP community. The Oracle USM lets you define two types of access levels: read-only and read-write. You can define multiple SNMP communities on a Oracle USM to segregate access modes per community and NMS host.

The members of an SNMP Community may be configured by IPv4, IPv6, or a combination of both in the **snmp-community** > **ip-address** parameter.

SNMP Trap Receiver Uses

A trap receiver is an application used to receive, log, and view SNMP traps for monitoring the Oracle USM (USM). An SNMP trap is the notification sent from a network device, such as the USM, that declares a change in service. Multiple trap receivers can be defined on an USM for either redundancy or to segregate alarms with different severity levels to individual trap receivers.

SNMP trap receivers may be configured with an IPv4 or IPv6 address in the **trap-receiver** > **ip-address** parameter.

Each Oracle Communications Session Delivery Manager which manages Oracle USMs should be configured on those SBCs as trap receivers.

IPv6 Trap Receiver Transport Support

This feature supports configuring trap receivers with IPv6 address notation, and allows traps to be sent to IPv6 targets.

Configuring SNMP

This section describes how to configure your Oracle USM to work with external SNMP systems. Sample configurations are also provided.

SNMP Configuration Overview

1. Configure the SNMP identification information. This step includes configuring the MIB system contact, name, and location parameters.
2. Set the general SNMP parameters to enable or disable SNMP on the Oracle USM. This step includes setting the switches that govern how the SNMP system responds to specified events.

3. Set the syslog events. This step includes setting the parameters for handling SNMP monitoring syslog events, which can trigger SNMP syslog traps.
4. Set SNMP communities. Configuration is separated into a unique configuration element.
5. Set trap receivers. Configuration is separated into a unique configuration element.

SNMP Configuration

To configure SNMP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **system-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

   From this point, you can set SNMP parameters. The following is an example what an SNMP configuration might look like. Parameters not described in this section are omitted below.

   ```
   system-config
           mib-system-contact             John Doe
           mib-system-name                Test System
           mib-system-location            Upstairs
           snmp-enabled                   enabled
           enable-snmp-auth-traps         disabled
           enable-snmp-syslog-notify      disabled
           enable-snmp-monitor-traps      disabled
           enable-env-monitor-traps       disabled
           snmp-syslog-his-table-length   1
           snmp-syslog-level              WARNING
   ```

System Wide Configuration for SNMP

This section describes the system-wide SNMP parameters found in the System Configuration element. These parameters set global SNMP information.

Set the following parameters to configure system wide SNMP functionality:

1. **mib-system-contact**—Set the contact information used within the system's MIB transactions. The SNMP agent sends this information to an NMS in response to an SNMP Get for the MIB-II sysContact MIB variable. This parameter's value can be a textual identification of your company's contact person for the system and information about how to contact that person.
2. **mib-system-name**—Set the identification of this Oracle USM presented within MIB transactions. This value, along with the target name of the system (identified in the boot parameters) are the values reported for MIB-II when an SNMP GET is issued by the NMS for the MIB-II sysName variable. This parameter has no direct relation to the hostname parameter in the system configuration element.

By convention, this is the node's FQDN. For SNMP MIB-II sysName GETs, the system returns SNMP communications in the following format: <targetName>[.<mib-system-name>]

targetName is the value configured in the target name (tn) boot parameter and mib-system-name is the value configured in this field.

3. **mib-system-location**—Set the physical location of this Oracle USM that is reported within MIB transactions. This parameter is reported when an SNMP GET is issued by the NMS for the MIB-II sysLocation variable. This parameter has no direct relation to the location field in the system configuration element.
4. **snmp-enabled**—Set the SNMP system on this Oracle USM to **enabled** or **disabled**. By default, this parameter is set to **enabled**. The valid values are:
   - enabled | disabled
5. **enable-snmp-syslog-notify**—Set whether SNMP traps are sent when the system generates an alarm message. The SNMP agent sends a trap when an alarm is generated if the following conditions are met:
   - SNMP is enabled.
   - This field is enabled.
   - The syslog severity level is equal to or greater than the severity level configured in the SNMP Syslog Level field.

     The default is **disabled**. Valid values are:
   - **enabled** | **disabled**
6. **enable-snmp-monitor-traps**—When this parameter is enabled, the Oracle USM generates traps with unique trap-IDs for each syslog event. If this parameter is disabled , a single trap-ID is used for all events, with different values in the description string. The default is **disabled**. The valid values are:
   - enabled | disabled
7. **enable-snmp-auth-traps**—Set whether the SNMP authentication traps are enabled. If an SNMP request fails authentication because of an IPv4 address and SNMP community mismatch, the SNMP request will be rejected. This field determines if an SNMP trap will be sent in response to the authentication failure. The default is **disabled**. Valid values for this parameter are:
   - **enabled** | **disabled**
8. **enable-env-monitor-traps**—Set whether or not the SNMP environment monitor traps are enabled. Environment traps include main board PROM temperature, CPU voltage, power supplies, fan speeds, etc. The default is **disabled**. Valid values for this parameter are:
   - enabled | disabled
9. **snmp-syslog-his-table-length**—Set the length of the syslog trap history table. When a syslog message that meets the SNMP syslog level field criteria is generated and SNMP is enabled, the SNMP agent adds that message to a history table. This parameter indicates the number of entries the table can contain. The default is **1**. The valid range is:
   - Minimum—1
   - Maximum—500

     Once the last table entry is filled, the oldest entry will be overwritten with a new entry.
10. **snmp-syslog-level**—Set the log severity level threshold that will cause the syslog trap to be sent to an NMS. When this criteria is met and the appropriate SNMP trap is sent, an

entry is written to the SNMP Syslog History Table. The default is **warning**. The following are valid values:

- emergency | critical | major | minor | warning | notice | info | trace | debug | detail

SNMP Community Configuration

To configure SNMP communities:

1. Access the **snmp-community** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# snmp-community
ORACLE(snmp-community)#
```

2. **community-name**—Set the SNMP community name of an active community where this Oracle USM can send or receive SNMP information. A community name value can also be used as a password to provide authentication, thereby limiting the NMSs that have access to this system. With this field, the SNMP agent provides trivial authentication based on the community name that is exchanged in plain text SNMP messages.
3. **access-mode**—Set the access level for all NMSs defined within this SNMP community. The access level determines the permissions that other NMS hosts can wield over this Oracle USM. The default is **read-only**. The valid values are:
   - **read-only**—allows GET requests.
   - **read-write**—unsupported.
4. **ip-addresses**—Set one or multiple IPv4 or IPv6 addresses that are valid within this SNMP community. These IP addresses correspond with the address of NMS applications that monitor or configure this Oracle USM. Include the addresses of all servers where Element Management Systems are installed.
5. Type **done** to save your configuration.

Trap Receiver Configuration

To configure trap receivers:

1. Access the **trap-receiver** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# trap-receiver
ORACLE(trap-receiver)#
```

2. Select the **trap-receiver** object to edit.

```
ORACLE(trap-receiver)# select
<ip-address>:

ORACLE(trap-receiver)#
```

3. **ip-address** — Set the IPv4 or IPv6 address of an authorized NMS. This parameter is the IPv4 or IPv6 address of an NMS where traps are sent. If you do not specify a port number, the default SNMP trap port of **162** will be used.
4. **filter-level** — Set the filter level threshold that indicates the severity level at which a trap is to be sent to this particular trap receiver. The default for this parameter is critical.

Example: When a trap with a severity level of **critical** is generated, the SNMP agent will send this trap only to NMSs that are configured in a trap-receiver element and have a **filter-level** value of **critical**.

The following table maps Syslog and SNMP alarms to trap receiver filter levels.

Filter Level	Syslog Severity Level	(SNMP) Alarm Severity Level
Critical	Emergency (1) Critical (2)	Emergency Critical
Major	Emergency (1) Critical (2) Major (3)	Emergency Critical Major
Minor	Emergency (1) Critical (2) Major (3) Minor (4)	Emergency Critical Major Minor
All	Emergency (1) Critical (2) Major (3) Minor (4) Warning (5) Notice (6) Info (7) Trace (8) Debug (9)	Emergency Critical Major Minor Warning

When configuring the trap-receiver element for use with Oracle Communications Session Element Manager systems, Oracle recommends that the value of **filter-level** be set to **All** for that configuration element that includes those servers.

5. **community-name** — Set the community name to which this trap receiver belongs. This community must be defined in the SNMP community element.
6. Type **done** to save your configuration.

Media Supervision Traps

The Oracle USM, when functioning as a border gateway, will send the following trap when the media supervision timer has expired. This behavior is disabled by default, but can be enabled by changing the **media-supervision-traps** parameter to **enabled** in the media-manager configuration element.

```
apSysMgmtMediaSupervisionTimerExpTrap    NOTIFICATION-TYPE
      OBJECTS          { apSysMgmtCallId }
      STATUS            current
      DESCRIPTION
            " The trap will be generated when a media supervision timer
            has expired.  This behavior is disabled by default but may
            be enabled by changing the 'media-supervision-traps'
            parameter of the 'media-manager' configuration element.  The
            included object is the call identifer for the call which had
            the timer expire."
      ::= { apSystemManagementMonitors 34 }
```

The system does not send this trap when functioning as an integrated Oracle USM.

Syslog and Process Logs

Logging events is a critical part of diagnosing misconfigurations and optimizing operations. Oracle USMs can send both syslog and process log data to appropriate hosts for storage and analysis.

Overview

The Oracle USM generates two types of logs, syslogs and process logs. Syslogs conform to the standard used for logging servers and processes as defined in RFC 3164.

Process logs are Oracle proprietary logs. Process logs are generated on a per-task basis and are used mainly for debugging purposes. Because process logs are more data inclusive than syslogs, their contents usually encompass syslog log data.

Syslog and process log servers are both identified by an IPv4 address and port pair.

Process Log Messages

Process log messages are sent as UDP packets in the following format:

```
<file-name>:<log-message>
```

In this format, <filename> indicates the log filename and <log-message> indicates the full text of the log message as it would appear if it were written to the normal log file.

Syslog and Process Logs Configuration

This section describes how to configure syslog and process log servers.

To configure syslogs and process logs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **system-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

   From this point, you can set process log parameters. Skip to the following process log configuration section.

4. Type **syslog-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual syslog parameters

   ORACLE(system-config)# **syslog-server**

   ORACLE(syslog-server)#

From this point, you can set syslog parameters. The following is an example what an syslog and process log configuration might look like. Parameters not described in this section are omitted below.

```
system-log-level              WARNING
syslog-server
        address                       172.15.44.12
        port                          514
        facility                      4
process-log-level             NOTICE
process-log-ip-address        0.0.0.0
process-log-port              0
```

Syslog Configuration

The Oracle USM supports multiple syslog servers. As the number of active syslog increases, the performance level of the Oracle USM may decrease. Therefore, we recommend configuring no more than 8 syslog servers.

Set the following parameters to configure syslog servers:

1. **address**—Set the IPv4 address of a syslog server.
2. **port**—Set the port portion of the syslog server. The default is **514**.
3. **facility**—Set an integer to identify a user-defined facility value sent in every syslog message from the Oracle USM to the syslog server. This parameter is used only for identifying the source of this syslog message as coming from the Oracle USM. It is not identifying an OS daemon or process. The default value for this parameter is **4**. RFC 3164 specifies valid facility values.

   In software release versions prior to Release 1.2, the Oracle USM would send all syslog messages with a facility marker of 4.
4. **system-log-level**—Set which log severity levels write to the system log (filename: acmelog). The default is **WARNING**. Valid values are:
   - EMERGENCY | CRITICAL | MAJOR | MINOR | WARNING | NOTICE | INFO | TRACE | DEBUG | DETAIL

Configure the Process Log Server

Set the following parameters to configure the process log server:

1. **process-log-level**—Set the starting log level all processes running on the system use. Each individual process running on the system has its own process log. The default is **NOTICE**. Valid values: EMERGENCY | CRITICAL | MAJOR | MINOR | WARNING | NOTICE | INFO | TRACE | DEBUG | DETAIL
2. **process-log-ip-address**—Set the IPv4 address of the process log server. The default value is **0.0.0.0**, which causes the system to write log messages to the normal log file.
3. **process-log-port**—Set the port number associated with the process log server. The default value is **0**, which causes the system to write log messages to the normal log file. The valid range is: 1025-65535.

Host Routes

Host routes let you insert entries into the Oracle USM's routing table. These routes affect traffic that originates at the Oracle USM's host process. Host routes are used primarily for steering management traffic to the correct network.

When traffic is destined for a network that is not explicitly defined on a Oracle USM, the default gateway (located in the **system-config**) is used. If you try to route traffic to a specific destination that is not accessible through the default gateway, you need to add a host route. Host routes can be thought of as a default gateway override.

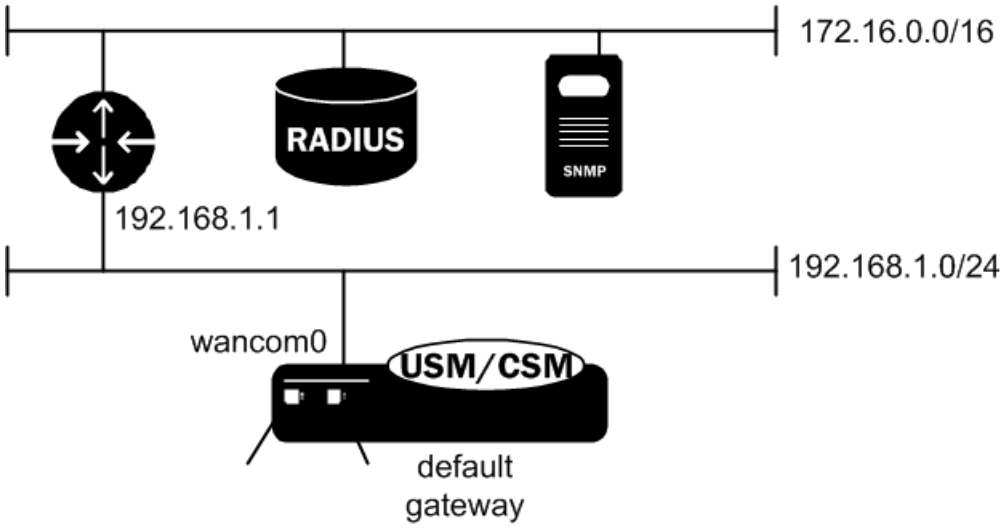
Certain SIP configurations require that the default gateway is located on a media interface. In this scenario, if management applications are located on a network connected to an administrative network, you will need to add a host route for management connectivity.

When source-based routing is used, the default gateway must exist on a media interface. Host routes might be needed to reach management applications connected to a management port in this kind of situation as well.

> **Note:**
>
> Do not configure a **host-route** > **gateway** with an address already used for any existing **network-interface** > **gateway**.

Host Routes Example

Because SIP signaling over media interfaces is enabled, the default gateway uses an IPv4 address assigned to a media interface. Maintenance services (SNMP and Radius) are located on a network connected to, but separate from, the 192.168.1.0/24 network on wancom0. In order to route Radius or SNMP traffic to an NMS (labeled as SNMP in the following example), a host route entry must be a part of the Oracle USM configuration. The host route tells the host how to reach the 172.16.0.0/16 network. The actual configuration is shown in the example in the next section of this guide.



Host Route Configuration

To configure a host route:

1. Access the **host-route** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# host-route
   ORACLE(host-route)#
   ```

2. **dest-network**—Set the IP address of the destination network that this host route points toward.
3. **netmask**—Set the netmask portion of the destination network for the route you are creating. The netmask is in dotted decimal notation.
4. **gateway**—Set the gateway that traffic destined for the address defined in the first two elements should use as its first hop.
5. Type **done** to save your configuration.

Setting Holidays in Local Policy

This section explains how to configure holidays on the Oracle USM.

You can define holidays that the Oracle USM recognizes. Holidays are used to identify a class of days on which a local policy is enacted. All configured holidays are referenced in the **local-policy-attributes** configuration subelement as an H in the **days-of-week** parameter. Because holidays are entered on a one-time basis per year, you must configure a new set of holidays yearly.

Holidays Configuration

To configure holidays:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-router-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-router-config
   ORACLE(session-router-config)#
   ```

4. Type **holidays** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router-config)# holidays
   ORACLE(session-router-holidays)#
   ```

   From this point, you can configure the holidays subelement. To view all holidays parameters, enter a **?** at the system prompt.

```
holiday
                date                            2005-01-01
                description                     New Years Day
```

To configure a holiday, add an entry for the following parameters in the holidays element:

5. **date**—Enter the holiday's date in YYYY-MM-DD format.
6. **description**—Enter a short description for the holiday you are configuring. If the description contains words separated by spaces, enter the full description surrounded by quotation marks.

Opening TCP Ports 3000 and 3001

This section explains how to open TCP ports 3000 and 3001 primarily for use with an element manager.

- TCP ports 3000 (used when notify commands are issued remotely, i.e. via an element management system) and 3001 (used for remote configuration, i.e. via an element management system), can be enabled or disabled in the system configuration

This configuration is not RTC enabled, so you must reboot your Oracle USM for changes to take effect.

Enable System to Connect to SDM

To control TCP ports 3000 and 3001 in the system configuration:

1. Access the **security-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# security-config
   ORACLE(security-config)#
   ```

2. Type **select** to begin editing the **system-config** object.

   ```
   ORACLE(system-config)# select
   ACMEPACKET(system-config)#
   ```

3. The parameter controlling ports 3000 and 3001 is called remote-control, and its default is enabled. To disable the ports, set this parameter to disabled.

   ```
   ORACLE(system-config)# remote-control disabled
   ```

4. Type **done** to save your configuration.
5. Reboot your Oracle USM. Type a **y** and press Enter to reboot.

   ```
   ORACLE# reboot
   ----------------------------------------
   WARNING: you are about to reboot this SD!
   ----------------------------------------
   Reboot this SD [y/n]?:y
   ```

DNS Transaction Timeout

This section explains how to configure the DNS transaction timeout interval on a per network-interface basis. You can currently configure the Oracle USM with a primary and two optional backup DNS servers. The Oracle USM queries the primary DNS server and upon not receiving

a response within the configured number of seconds, queries the backup1 DNS server and if that times out as well, then contacts the backup2 DNS server.

Retransmission Logic

The retransmission of DNS queries is controlled by three timers. These timers are derived from the configured DNS timeout value and from underlying logic that the minimum allowed retransmission interval should be 250 milliseconds; and that the Oracle USM should retransmit 3 times before timing out to give the server a chance to respond.

- Init-timer is the initial retransmission interval. If a response to a query is not received within this interval, the query is retransmitted. To safeguard from performance degradation, the minimum value allowed for this timer is 250 milliseconds.
- Max-timer is the maximum retransmission interval. The interval is doubled after every retransmission. If the resulting retransmission interval is greater than the value of max-timer, it is set to the max-timer value.
- Expire-timer: is the query expiration timer. If a response is not received for a query and its retransmissions within this interval, the server will be considered non-responsive and the next server in the list will be tried.

The following examples show different timeout values and the corresponding timers derived from them.

```
timeout >= 3 seconds
Init-timer = Timeout/11
Max-Timer = 4 * Init-timer
Expire-Timer = Timeout
timeout = 1 second
Init-Timer = 250 ms
Max-Timer = 250 ms
Expire-Timer = 1 sec
timeout = 2 seconds
Init-Timer = 250 ms
Max-Timer = 650 ms
Expire-Timer = 2sec
```

DNS Transaction Timeout Configuration

To configure DNS transaction timeout:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **network-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# network-interface
   ORACLE(network-interface)#
   ```

   From this point, you can configure network interface parameters. To view all network interface parameters, enter a **?** at the system prompt.

4. **dns-timeout**—Enter the total time in seconds you want to elapse before a query (and its retransmissions) sent to a DNS server would timeout. The default is **11** seconds. The valid range is:
   - Minimum—1
   - Maximum—999999999.

     If a query sent to the primary DNS server times out, the backup1 DNS server is queried. If the query times out after the same period of time elapses, the query continues on to the backup2 DNS server.
5. Save and activate your configuration.

DNS Server Status via SNMP

The Oracle USM monitors the status of all configured DNS servers used by a SIP daemon. If a DNS server goes down, a major alarm is sent. If all DNS servers used by a SIP daemon are down, a critical alarm is sent. The apAppsDnsServerStatusChangeTrap is sent for both events.

You can poll the status of a DNS server using the apAppsDNSServerStatusTable in the ap-apps.mib.

Once the apAppsDnsServerStatusChangeTrap has been sent, a 30 second window elapses until the server status is checked again. At the 30 second timer expiration, if the server is still down, another trap and alarm are sent. If the server has been restored to service, the apAppsDnsServerStatusChangeClearTrap is sent.

Persistent Protocol Tracing

This section explains how to configure persistent protocol tracing to capture specific SIP protocol message logs and persistently send them off the Oracle USM, even after rebooting the system. This feature is not applicable to log for H.323 or IWF.

About Persistent Protocol Tracing

You can configure sending protocol message logs off of the Oracle USM, and have that persist after a reboot. You no longer have to manually issue the notify command each time you reboot.

To support persistent protocol tracing, you configure the following system-config parameters:

- **call-trace**—Enable/disable protocol message tracing (currently only sipmsg.log and alg.log) regardless of the process-log-level setting. If the process-log-level is set to trace or debug, call-trace will not disable.
- **internal-trace**—Enable/disable internal ACP message tracing for all processes, regardless of process-log-level setting. This applies to all *.log (internal ACP message exchange) files other than sipmsg.log and alg.log. If the process-log-level is set to trace or debug, call-trace will not disable.
- **log-filter**—Determine what combination of protocol traces and logs are sent to the log server defined by the process-log-ip parameter value. You can also fork the traces and logs, meaning that you keep trace and log information in local storage as well as sending it to the server. You can set this parameter to any of the following values: none, traces, traces-fork, logs, logs, all, or all-fork.
  The Oracle USM uses the value of this parameter in conjunction with the process-log-ip and process-log-port values to determine what information to send. If you have configured the proc-log-ip and proc-log-port parameters, choosing traces sends just the trace

information (provided they are turned on), logs sends only process logs (log.*), and all sends everything (which is the default).

About the Logs

When you configure persistent protocol tracing, you affect the following types of logs.

> **Note:**
>
> Enabling logs can have an impact on Oracle USM performance.

Process Logs

Events are logged to a process log flow from tasks and are specific to a single process running on the Oracle USM. By default they are placed into individual files associated with each process with the following name format:

log.<taskname>

By setting the new log-filter parameter, you can have the logs sent to a remote log server (if configured). If you set log-filter to logs or all, the logs are sent to the log server. Otherwise, the logs are still captured at the level the process-log-level parameter is set to, but the results are stored on the Oracle USM's local storage.

Communication Logs

These are the communication logs between processes and system management. The logs are usually named <name>.log, with <name> being the process name. For example, sipd.log.

This class of log is configured by the new internal-trace parameter.

Protocol Trace Logs

The only protocol trace logs included at this time are sipmsg.log for SIP. The H.323 system tracing is not included. All of the logs enabled with the call–trace parameter are sent to remote log servers, if you also set the log-filter parameter to logs or all.

Persistent Protocol Tracing Configuration

Before you configure persistent protocol tracing, ensure you have configured the process logs by setting the system configuration's **process-log-ip** parameter.

To configure persistent protocol tracing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **system-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(system)# system-config
ORACLE(system-config)#
```

4. **call-trace**—Set to **enabled** to enable protocol message tracing for sipmsg.log for SIP. The default is **disabled**. The valid values are:
   - enabled | disabled
5. **internal-trace**—Set to **enabled** to enable internal ACP message tracing for all processes. The default is **disabled**. The valid values are:
   - enabled | disabled
6. **log-filter**—Choose the appropriate setting for how you want to send and/or store trace information and process logs. The valid values are:
   - **none**—No information will be sent or stored.
   - traces—Sends the trace information to both the log server; includes <name>.log files that contain information about the Oracle USM's internal communication processes (<name> is the name of the internal process)
   - **traces-fork**—Sends the trace information to both the log server and also keeps it in local storage; includes <name>.log files that contain information about the Oracle USM's internal communication processes (<name> is the name of the internal process)
   - **logs—**Sends the process logs to both the log server; includes log.* files, which are Oracle USM process logs
   - **logs-fork**—Sends the process logs to both the log server and also keeps it in local storage; includes log.* files, which are Oracle USM process logs
   - **all—**Sends all logs to the log servers that you configure
   - **all-fork**—Sends all logs to the log servers that you configure, and it also keeps the logs in local storage
7. Save and activate your configuration.

System Access Control

You can configure a system access control list (ACL) for your Oracle USM that determines what traffic the Oracle USM allows over its management interface (wancom0). By specifying who has access to the Oracle USM via the management interface, you can provide DoS protection for this interface.

Using a list of IP addresses and subnets that are allowable as packet sources, you can configure what traffic the Oracle USM accepts and what it denies. All IP packets arriving on the management interface are subject; if it does not match your configuration for system ACL, then the Oracle USM drops it.

> **Note:**
>
> All IP addresses configured in the SNMP community table are automatically permitted.

Adding an ACL for the Management Interface

The new subconfiguration **system-access-list** is now part of the system configuration, and its model is similar to host routes. For each entry, you must define an IP destination address and mask; you can specify either the individual host or a unique subnet.

If you do not configure this list, then there will be no ACL/DoS protection for the Oracle USM's management interface.

You access the **system-access-list** via system path, where you set an IP address and netmask. You can configure multiple system ACLs using this configuration.

To add an ACL for the management interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type system-access-list and press Enter.

   ```
   ORACLE(system)# system-access-list
   ORACLE(system-access-list)#
   ```

4. **source-address**—Enter the IP address representing for the source network for which you want to allow traffic over the management interface.

5. **netmask**—Enter the netmask portion of the source network for the traffic you want to allow. The netmask is in dotted decimal notation.

Notes on Deleting System ACLs

If you delete a system ACL from your configuration, the Oracle USM checks whether or not there are any active FTP or Telnet client was granted access when the entry was being removed. If such a client were active during ACL removal, the Oracle USM would warn you about the condition and ask you to confirm the deletion. If you confirm the deletion, then the Oracle USM's session with the active client is suspended.

The following example shows you how the warning message and confirmation appear. For this example, and ACLI has been deleted, and the user is activating the configuration that reflects the change.

```
ORACLE # activate-config
Object deleted will cause service disruption:
 system-access-list: identifier=172.30.0.24
 ** WARNING: Removal of this system-ACL entry will result
             in the lockout of a current FTP client
Changes could affect service, continue (y/n) y
Activate-Config received, processing.
```

System TCP Keepalive Settings

You can configure the Oracle USM to control TCP connections by setting:

- The amount of time the TCP connection is idle before the Oracle USM starts sending keepalive messages to the remote peer
- The number of keepalive packets the Oracle USM sends before terminating the TCP connection

If TCP keepalive fails, then the Oracle USM will drop the call associated with that TCP connection.

In the ALCI, a configured set of network parameters appears as follows:

```
network-parameters
        tcp-keepinit-timer              75
        tcp-keepalive-count             4
        tcp-keepalive-idle-timer        400
        tcp-keepalive-interval-timer    75
        tcp-keepalive-mode              0
```

Then you apply these on a per-interface basis.

System TCP Keepalive Configuration

TCP setting are global, and then enabled or disabled on a per-interface basis.

To configure TCP keepalive parameters on your Oracle USM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-related configurations.

   ```
   ORACLE(configure)# system
   ```

3. Type **network-parameters** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **tcp-keepinit-timer—If a TCP connection cannot be established within some amount of time, TCP will time out the connect attempt. It can be used to set the initial timeout period for a given socket, and specifies the number of seconds to wait before the connect attempt is timed out. For passive connections, this value is inherited from the listening socket. The default is 75. The valid range is:**

   - **Minimum—0**
   - **Maximum—999999999.**

5. **tcp-keepalive-count**—Enter the number of packets the Oracle USM sends to the remote peer before it terminates the TCP connection. The default is **8**. The valid range is:

   - Minimum—0
   - Maximum—223-1

6. **tcp-keepalive-idle-timer**—Enter the number of seconds of idle time before TCP keepalive messages are sent to the remote peer if the **SO-KEEPALIVE** option is set. The default is **7200**. The valid range is:

   - Minimum—30
   - Maximum—7200

7. **tcp-keepalive-interval-timer**—When the SO_KEEPALIVE option is enabled, TCP probes a connection that has been idle for some amount of time. If the remote system does not respond to a keepalive probe, TCP retransmits the probe after a set amount of time. This parameter specifies the number of seconds to wait before retransmitting a keepalive probe. The default value is **75** seconds. **The valid range is:**
   - Minimum—15
   - Maximum—75
8. **tcp-keepalive-mode**—Set the TCP keepalive response sequence number. The default is **0**. The valid values are:
   - 0—The sequence number is sent un-incremented
   - 1—The number is incremented
   - 2—No packets are sent

Configurable TCP Timers

You can configure your Oracle USM to detect failed TCP connections more quickly so that data can be transmitted via an alternate connection before timers expire. Across all protocols, you can now control the following for TCP:

- Connection establishment
- Data retransmission
- Timer for idle connections

These capabilities all involve configuring an **options** parameter that appears in the network parameters configuration.

Configuring TCP Connection Establishment

To establish connections, TCP uses a three-way handshake during which two peers exchange TCP SYN messages to request and confirm the active open connection. In attempting this connection, one peer retransmits the SYN messages for a defined period of time if it does not receive acknowledgement from the terminating peer. You can configure the amount of time in seconds between the retries as well as how long (in seconds) the peer will keep retransmitting the messages.

You set two new options in the network parameters configuration to specify these amounts of time: **atcp-syn-rxmt-interval** and **atcp-syn-rxmt-maxtime**.

Note that for all configured options, any values entered outside of the valid range are silently ignored during configuration and generate a log when you enter the **activate** command.

To configure TCP connection establishment:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **network-parameters** and press Enter.

```
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-syn-rxmt-interval=x** (where x is a value in seconds between 2 and 10) with a plus sign in front of it. Then press Enter. This value will be used as the interval between TCP SYN messages when the Oracle USM is trying to establish a connection with a remote peer.

   Now enter a second option to set the maximum time for trying to establish a TCP connection. Set the options parameter by typing **options**, a Space, the option name **atcp-syn-rxmt-maxtime=x** (where x is a value in seconds between 5 and 75) with a plus sign in front of it. Then press Enter.

   ```
   ORACLE(network-parameters)# options +atcp-syn-rxmt-interval=5
   ORACLE(network-parameters)# options +atcp-syn-rxmt-maxtime=30
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

   > **Note:**
   >
   > **atcp-syn-rxmt-maxtime=x option is equivalent to the tcp-keepinit-timer parameter, but only affects ATCP.**

5. Save and activate your configuration.

Configuring TCP Data Retransmission

TCP is considered reliable in part because it requires that entities receiving data must acknowledge transmitted segments. If data segments go unacknowledged, then they are retransmitted until they are finally acknowledged or until the maximum number of retries has been reached. You can control both the number of times the Oracle USM tries to retransmit unacknowledged segments and the periodic interval (how often) at which retransmissions occur.

You set two new options in the network parameters configuration to specify how many retransmissions are allowed and for how long: **atcp-rxmt-interval** and **atcp-rxmt-count**.

To configure TCP data retransmission:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-rxmt-interval=x** (where x is a value in seconds between 2 and 60) with a plus sign in front

of it. Then press Enter. This value will be used as the interval between retransmission of TCP data segments that have not been acknowledged.

Now enter a second option to set the number of times the Oracle USM will retransmit a data segment before it declares the connection failed. Set the options parameter by typing **options**, a Space, the option name **atcp-rxmt-count=x** (where x is a value between 4 and 12 representing how many retransmissions you want to enable) with a plus sign in front of it. Then press Enter.

```
ORACLE(network-parameters)# options +atcp-rxmt-interval=30
ORACLE(network-parameters)# options +atcp-rxmt-count=6
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Timer for Idle Connections

When enabled to do so, the Oracle USM monitors inbound TCP connections for inactivity. These are inbound connections that the remote peer initiated, meaning that the remote peer sent the first SYN message. You can configure a timer that sets the maximum amount of idle time for a connection before the Oracle USM consider the connection inactive. Once the timer expires and the connection is deemed inactive, the Oracle USM sends a TCP RST message to the remote peer.

To configure the timer for TCP idle connections:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **atcp-idle-timer=x** (where x is a value in seconds between 120 and 7200) with a plus sign in front of it. Then press Enter. This value will be used to measure the activity of TCP connections; when the inactivity on a TCP connection reaches this value in seconds, the Oracle USMdeclares it inactive and drops the session.

   ```
   ORACLE(network-parameters)# options +atcp-idle-timer=900
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Historical Data Recording (HDR)

Updated HDR and HDR configuration information resides in the Oracle Communications C-Series Historical Data Recording (HDR) Resource Guide Version C6.3.0, 400-0141-63 (Oracle Communications S-CX6.3.0 HDR Resource Guide.pdf). This document is available with the complete S-CX6.3.0 documentation set.

RAMdrive Log Cleaner

The RAMdrive log cleaner allows the Oracle USM to remove log files proactively and thereby avoid situations where running low on RAMdrive space is a danger. Because even a small amount of logging can consume a considerable space, you might want to enable the RAMdrive log cleaner.

The RAMdrive cleaner periodically checks the remaining free space in the RAMdrive and, depending on the configured threshold, performs a full check on the /ramdrv/logs directory. During the full check, the RAMdrive cleaner determines the total space logs files are using and deletes log files that exceed the configured maximum lifetime. In addition, if the cleaner finds that the maximum log space has been exceeded or the minimum free space is not sufficient, it deletes older log files until the thresholds are met.

Not all log files, however, are as active as others. This condition affects which log files the log cleaner deletes to create more space in RAMdrive. More active log files rotate through the system more rapidly. So, if the log cleaner were to delete the oldest of these active files, it might not delete less active logs files that could be older than the active ones. The log cleaner thus deletes files that are truly older, be they active or inactive.

Applicable Settings

In the system configuration, you establish a group of settings in the options parameter that control the log cleaner's behavior:

- **ramdrv-log-min-free**—Minimum percent of free space required when rotating log files. When the amount of free space on the RAMdrive falls below this value, the log cleaner deletes the oldest copy of the log file. The log cleaner also uses this setting when performing period cleaning.
- **ramdrv-log-max-usage**—Maximum percent of the RAMdrive the log files can use. The log cleaner removes old log files to maintain this threshold.
- **ramdrv-log-min-check**—Minimum percent of free space on the RAMdrive that triggers the log cleaner to perform a full check of log files.
- **ramdrv-min-log-check**—Minimum time (in seconds) between log cleaner checks.
- **ramdrv-max-log-check**—Maximum time (in seconds) between log cleaner checks. This value must be greater than or equal to the **ramdrv-min-log-check**.
- **ramdrv-log-lifetime**—Maximum lifetime (in days) for log files. You give logs unlimited lifetime by entering a value of 0.

Clean-Up Procedure

The log cleaner checks the amount of space remaining in the RAMdrive and performs a full check of the logs directory when:

- Free space is less than the minimum percent of the RAMdrive that triggers a full check of log files
- The amount of free space has changed by more than 5% of the RAMdrive capacity since the last full check
- A full check of the logs directory has not been performed in the last hour

When it checks the logs directory, the log cleaner inventories the collected log files. It identifies each files as one of these types:

- Process log—Files beginning with log.
- Internal trace file—A <task>.log file
- Protocol trace file—Call trace including sipmsg.log, dns.log, sipddns.log, and alg.log
- CDR file—File beginning with cdr

Next, the log cleaner determines the age of the log files using the number of seconds since the log files were created. Then it orders the files from oldest to newest. The age adjusts such that it always increases as the log file sequence number (a suffix added by file rotation) increases. The log cleaner applies an additional weighting factor to produce a weighted age that favors the preservation of protocol traces files over internal trace files, and internal trace files over process log files. The base log file and CDR files are excluded from the age list and so will not be deleted; the accounting configuration controls CDR file aging.

With the age list constructed, the log cleaner examines the list from highest weighted age to lowest. If the actual file age exceeds the RAMdrive maximum log lifetime, the log cleaner deletes it. Otherwise, the log cleaner deletes files until the maximum percent of RAMdrive that logs can use is no longer exceeded and until the minimum percent of free space required when rotating logs is available.

Clean-Up Frequency

The minimum free space that triggers a full check of log files and the maximum time between log file checks control how often the log cleaner performs the clean-up procedure. When it completes the procedure, the log cleaner determines the time interval until the next required clean-up based on the RAMdrive's state.

If a clean-up results in the deletion of one or more log files or if certain thresholds are exceeded, frequency is based on the minimum time between log cleaner checks. Otherwise, the system gradually increases the interval up to the maximum time between log cleaner checks. The system increases the interval by one-quarter of the difference between the minimum and maximum interval, but not greater than one-half the minimum interval or smaller than 10 seconds. For example, using the default values, the interval would be increased by 30 seconds.

RAMdrive Log Cleaner Configuration

You configure the log cleaner's operating parameters and thresholds in the system configuration. Note that none of these settings is RTC-supported, so you must reboot your Oracle USM in order for them to take effect. If you are using this feature on an HA node, however, you can add this feature without impact to service by activating the configuration, rebooting the standby, switching over to make the newly booted standby active, and then rebooting the newly standby system.

Unlike other values for **options** parameters, the Oracle USM validates these setting when entered using the ACLI. If any single value is invalid, they all revert to their default values.

To configure the RAMdrive log cleaner:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **system** and press Enter.

```
ORACLE(configure)# system
ORACLE(system)#
```

3. Type **system-config** and press Enter.

```
ORACLE(system)# system-config
ORACLE(system-config)#
```

4. **options**—Set the options parameter by typing options, a Space, **<option name>=X (where X is the value you want to use)** with a plus sign in front of it. Then press Enter.

   Remember that if any of your settings are invalid, the Oracle USM changes the entire group of these options back to their default settings.

Option Name	Description
ramdrv-log-min-free	Minimum percent of free space required when rotating log files. When the amount of free space on the RAMdrive falls below this value, the log cleaner deletes the oldest copy of the log file. The log cleaner also uses this setting when performing period cleaning. Default=40; Minimum=15; Maximum=75
ramdrv-log-max-usage	Maximum percent of the RAMdrive the log files can use. The log cleaner removes old log files to maintain this threshold. Default=40; Minimum=15; Maximum=75
ramdrv-log-min-check	Minimum percent of free space on the RAMdrive that triggers the log cleaner to perform a full check of log files. Default=50; Minimum=25; Maximum=75
ramdrv-min-log-check	Maximum time (in seconds) between log cleaner checks. This value must be greater than or equal to the ramdrv-min-log-check. Default=180; Minimum=40; Maximum=1800
ramdrv--log-lifetime	Maximum lifetime (in days) for log files. You give logs unlimited lifetime by entering a value of 0. Default=30; Minimum=2; Maximum=9999

```
ORACLE(system-config)# options +ramdrv-log-min-free=50
ORACLE(system-config)# options +ramdrv-log-max-usage=50
ORACLE(system-config)# options +ramdrv-log-min-check=35
ORACLE(system-config)# options +ramdrv-min-log-check=120
ORACLE(system-config)# options +ramdrv-max-log-free=1500
ORACLE(system-config)# options +ramdrv-log-lifetime=7
```

   If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Reboot your Oracle USM.

Configurable Alarm Thresholds and Traps

The Oracle USM supports user-configurable threshold crossing alarms. These configurations let you identify system conditions of varying severity which create corresponding alarms of varying severity. You configure an alarm threshold type which indicates the resource to monitor. The available types are:

- cpu — CPU utilization monitored as a percentage of total CPU capacity
- memory — memory utilization monitored as a percentage of total memory available

> **Note:**
>
> When you configure an **alarm-threshold** for **memory** with **severity** set to **critical**, the Oracle USM will stop processing traffic if that configured value is reached, regardless of how low the value is.

- sessions — license utilization monitored as a percentage of licensed session capacity
- space — remaining disk space (configured in conjunction with the volume parameter - see the Storage Expansion Module Monitoring section of the *Accounting Guide* for more information.)
- deny-allocation — denied entry utilization monitored as a percentage of reserved, denied entries.

For the alarm type you create, the Oracle USM can monitor for 1 through 3 severity levels as minor, major, and critical. Each of the severities is configured with a corresponding value that triggers that severity. For example the configuration for a CPU alarm that is enacted when CPU usage reaches 50%:

```
alarm-threshold
        type                            cpu
        severity                        minor
        value                           50
```

You may create addition CPU alarms for increasing severities. For example:

```
alarm-threshold
        type                            cpu
        severity                        critical
        value                           90
```

The alarm state is enacted when the resource defined with the type parameter exceeds the value parameter. When the resource drops below the value parameter, the alarm is cleared.

SNMP Traps

When a configured alarm threshold is reached, the Oracle USM sends an apSysMgmtGroupTrap. This trap contains the resource type and value for the alarm configured in the alarm-threshold configuration element. The trap does not contain information associated with configured severity for that value.

```
apSysMgmtGroupTrap          NOTIFICATION-TYPE
    OBJECTS         { apSysMgmtTrapType, apSysMgmtTrapValue }
    STATUS            current
```

```
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          exceeds a certain threshold. "
    ::= { apSystemManagementNotifications 1 }
```

When the resource usage retreats below a configured threshold, the Oracle USM sends an apSysMgmtGroupClearTrap.

```
apSysMgmtGroupClearTrap         NOTIFICATION-TYPE
    OBJECTS        { apSysMgmtTrapType }
    STATUS           current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          returns to within a certain threshold.  This signifies that
          an alarm caused by that monitoring object has been cleared. "
    ::= { apSystemManagementNotifications 2 }
```

The alarm and corresponding traps available through the User Configurable Alarm Thresholds functionality are summarized in the following table.

Alarm	Severity	Cause	Actions
CPU	minor major	high CPU usage	apSysMgmtGroupTrap sent with apSysCPUUtil
	critical		apSysMgmtTrapValue
memory	minor major	high memory usage	apSysMgmtGroupTrap sent with apSysMemoryUtil
	critical		apSysMgmtTrapValue
sessions	minor major	high license usage	apSysMgmtGroupTrap sent with apSysLicenseCapacity
	critical		apSysMgmtTrapValue
space	minor major	high HDD usage, per volume	apSysMgmtStorageSpaceAvailThresholdTrap sent with:
	critical		apSysMgmtSpaceAvailCurrent
			apSysMgmtSpaceAvailMinorThreshold
			apSysMgmtSpaceAvailMajorThreshold
			apSysMgmtSpaceAvailCriticalThreshold
			apSysMgmtPartitionPath
deny allocation	minor major	high usage of denied ACL entries	apSysMgmtGroupTrap sent with apSysCurrentEndptsDenied
	critical		apSysMgmtTrapValue

Alarm Thresholds Configuration

To configure alarm thresholds:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type system and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type system-config and press Enter.

```
ORACLE(system)# system-config
ORACLE(system-config)#
```

4. Type alarm-threshold and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(system-config)# alarm-threshold
ORACLE(alarm-threshold)#
```

5. type — Enter the type of resource which this alarm monitors. Valid values include:
   - cpu
   - memory
   - sessions
   - space
   - deny-allocation
6. volume — Enter the logical disk volume this alarm monitors (used only in conjunction when type = space).
7. severity — Set the severity of the threshold. Valid values include:
   - minor
   - major
   - critical
8. value — Enter the value from 1 to 99, indicating the percentage, which when exceeded generates an alarm.
9. Save and activate your configuration.

Alarm Synchronization

Two trap tables in the ap-smgmt.mib record trap information for any condition on the Oracle USM that triggers an alarm condition. You can poll these two tables from network management systems, OSS applications, and the Session Delivery Manager to view the fault status on one or more Oracle USM s.

The two trap tables that support alarm synchronization, and by polling them you can obtain information about the current fault condition on the Oracle USM . These tables are:

- apSysMgmtTrapTable—You can poll this table to obtain a summary of the Oracle USM 's current fault conditions. The table records multiples of the same trap type that have occurred within a second of one another and have different information. Each table entry contains the following:
  - Trap identifier
  - System time (synchronized with an NTP server)
  - sysUpTime
  - Instance number
  - Other trap information for this trap identifier
- apSysMgmtTrapInformationTable—You can poll this table to obtain further details about the traps recorded in the apSysMgmtTrapTable table. The following information appears:
  - Data index

- Data type
- Data length
- The data itself (in octets)

Trap tables do not record information about alarm severity.

The apSysMgmtTrapTable can hold up to 1000 entries, and you can configure the number of days these entries stay in the table for a maximum of seven days. If you set this parameter to 0 days, the feature is disabled. And if you change the setting to 0 days from a greater value, then the Oracle USM purges the tables.

Caveats

Note that the Oracle USM does not replicate alarm synchronization table data across HA nodes. That is, each Oracle USM in an HA node maintains its own tables.

Alarm Synchronization Configuration

You turn on alarm synchronization in the system configuration.

To use alarm synchronization:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **system-config** and press Enter.

   ```
   ORACLE(system)# system-config
   ORACLE(system-config)#
   ```

4. **trap-event-lifetime**—To enable alarm synchronization—and cause the Oracle USM to record trap information in the apSysMgmtTrapTable and the apSysMgmtTrapInformationTable—set this parameter to the number of days you want to keep the information. Leaving this parameter set to 0 (default) turns alarm synchronization off, and you can keep information in the tables for up to 7 days. 7 is the maximum value for this parameter.

Accounting Configuration

The Oracle USM offers support for RADIUS, an accounting, authentication, and authorization (AAA) system. In general, RADIUS servers are responsible for receiving user connection requests, authenticating users, and returning all configuration information necessary for the client to deliver service to the user.

You can configure your Oracle USM to send call accounting information to one or more RADIUS servers. This information can help you to see usage and QoS metrics, monitor traffic, and even troubleshoot your system.

This guide contains all RADIUS information, as well as information about:

- Local CDR storage on the Oracle USM , including CSV file format settings

- The ability to send CDRs via FTP to a RADIUS sever (the FTP push feature)
- Per-realm accounting control
- Configurable intermediate period
- RADIUS CDR redundancy
- RADIUS CDR content control

Stream Control Transfer Protocol Overview

The Stream Control Transmission Protocol (SCTP) was originally designed by the Signaling Transport (SIGTRAN) group of IETF for Signalling System 7 (SS7) transport over IP-based networks. It is a reliable transport protocol operating on top of an unreliable connectionless service, such as IP. It provides acknowledged, error-free, non-duplicated transfer of messages through the use of checksums, sequence numbers, and selective retransmission mechanism.

SCTP is designed to allow applications, represented as endpoints, communicate in a reliable manner, and so is similar to TCP. In fact, it has inherited much of its behavior from TCP, such as association (an SCTP peer-to-peer connection) setup, congestion control and packet-loss detection algorithms. Data delivery, however, is significantly different. SCTP delivers discrete application messages within multiple logical streams within the context of a single association. This approach to data delivery is more flexible than the single byte-stream used by TCP, as messages can be ordered, unordered or even unreliable within the same association.

SCTP Packets

SCTP packets consist of a common header and one or more chunks, each of which serves a specific purpose.

- DATA chunk — carries user data
- INIT chunk — initiates an association between SCTP endpoints
- INIT ACK chunk — acknowledges association establishment
- SACK chunk — acknowledges received DATA chunks and informs the peer endpoint of gaps in the received subsequences of DATA chunks
- HEARTBEAT chunk — tests the reachability of an SCTP endpoint
- HEARTBEAT ACK chunk — acknowledges reception of a HEARTBEAT chunk
- ABORT chunk — forces an immediate close of an association
- SHUTDOWN chunk — initiates a graceful close of an association
- SHUTDOWN ACK chunk — acknowledges reception of a SHUTDOWN chunk
- ERROR chunk — reports various error conditions
- COOKIE ECHO chunk — used during the association establishment process
- COOKIE ACK chunk — acknowledges reception of a COOKIE ECHO chunk
- SHUTDOWN COMPLETE chunk — completes a graceful association close

SCTP Terminology

This section defines some terms commonly found in SCTP standards and documentation.

SCTP Association

is a connection between SCTP endpoints. An SCTP association is uniquely identified by the transport addresses used by the endpoints in the association. An SCTP association can be represented as a pair of SCTP endpoints, for example, assoc = { [IPv4Addr : PORT1], [IPv4Addr1, IPv4Addr2: PORT2]}.

Only one association can be established between any two SCTP endpoints.

SCTP Endpoint

is a sender or receiver of SCTP packets. An SCTP endpoint may have one or more IP address but it always has one and only one SCTP port number. An SCTP endpoint can be represented as a list of SCTP transport addresses with the same port, for example, endpoint = [IPv6Addr, IPv6Addr: PORT].

An SCTP endpoint may have multiple associations.

SCTP Path

is the route taken by the SCTP packets sent by one SCTP endpoint to a specific destination transport address or its peer SCTP endpoint. Sending to different destination transport addresses does not necessarily guarantee separate routes.

SCTP Primary Path

is the default destination source address, the IPv4 or IPv6 address of the association initiator. For retransmissions however, another active path may be selected, if one is available.

SCTP Stream

is a unidirectional logical channel established between two associated SCTP endpoints. SCTP distinguishes different streams of messages within one SCTP association. SCTP makes no correlation between an inbound and outbound stream.

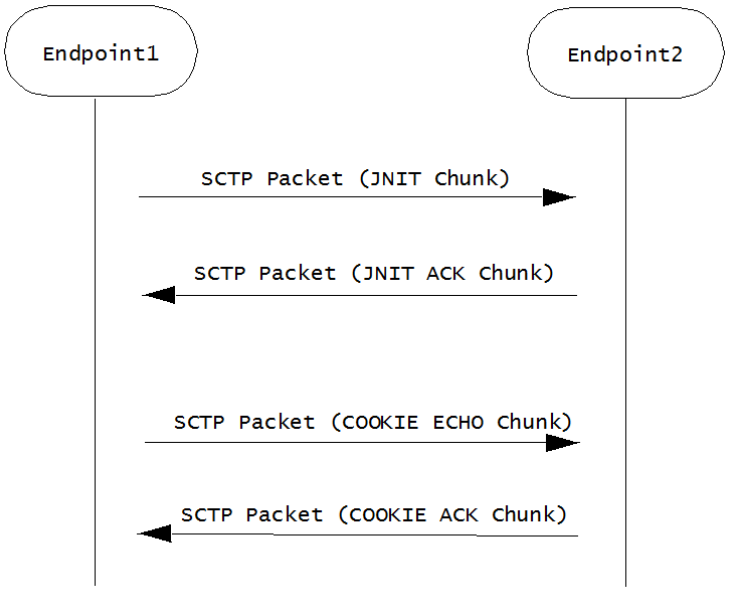
SCTP Transport Address

is the combination of an SCTP port and an IP address. For the current release, the IP address portion of an SCTP Transport Address must be a routable, unicast IPv4 or IPv6 address.

An SCTP transport address binds to a single SCTP endpoint.

SCTP Message Flow

Before peer SCTP users (commonly referred to as endpoints) can send data to each other, an association (an SCTP connection) must be established between the endpoints. During the association establishment process a cookie mechanism is employed to provide protection against security attacks. The following figure shows a sample SCTP association establishment message flow.



Endpoint1 initiates the association by sending Endpoint2 an SCTP packet that contains an INIT chunk, which can include one or more IP addresses used by the initiating endpoint. Endpoint2 acknowledges the initiation of an SCTP association with an SCTP packet that contains an INIT_ACK chunk. This chunk can also include one or more IP addresses at used by the responding endpoint.

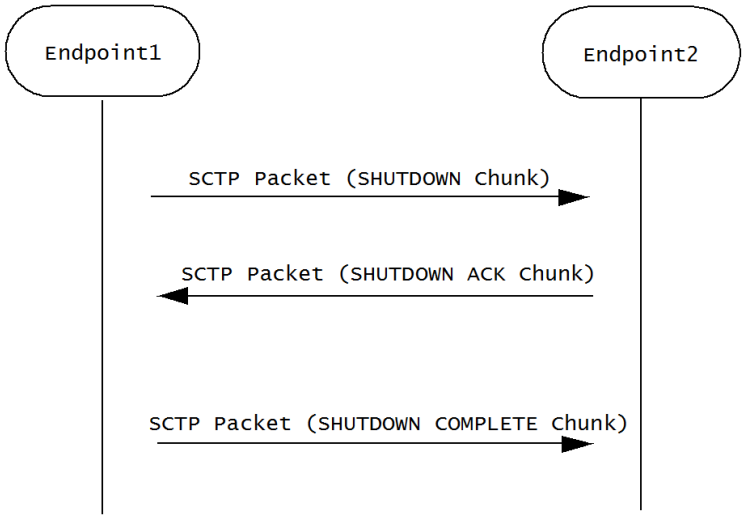
Both the INIT chuck (issued by the initiator) and INIT ACK chunk (issued by the responder) specify the number of outbound streams supported by the association, as well as the maximum inbound streams accepted from the other endpoint.

Association establishment is completed by a COOKIE ECHO/COOKIE ACK exchange that specifies a cookie value used in all subsequent DATA exchanges.

Once an association is successfully established, an SCTP endpoint can send unidirectional data streams using SCTP packets that contain DATA chunks. The recipient endpoint acknowledges with an SCTP packet containing a SACK chunk.

SCTP monitors endpoint reachability by periodically sending SCTP packets that contain HEARTBEAT chunks. The recipient endpoint acknowledges receipt, and confirms availability, with an SCTP packet containing a HEARBEAT ACK chunk.

Either SCTP endpoint can initiate a graceful association close with an SCTP packet that contains a SHUTDOWN chunk. The recipient endpoint acknowledges with an SCTP packet containing a SHUTDOWN ACK chunk. The initiating endpoint concludes the graceful close with an SCTP packet that contains a SHUTDOWN COMPLETE chunk.

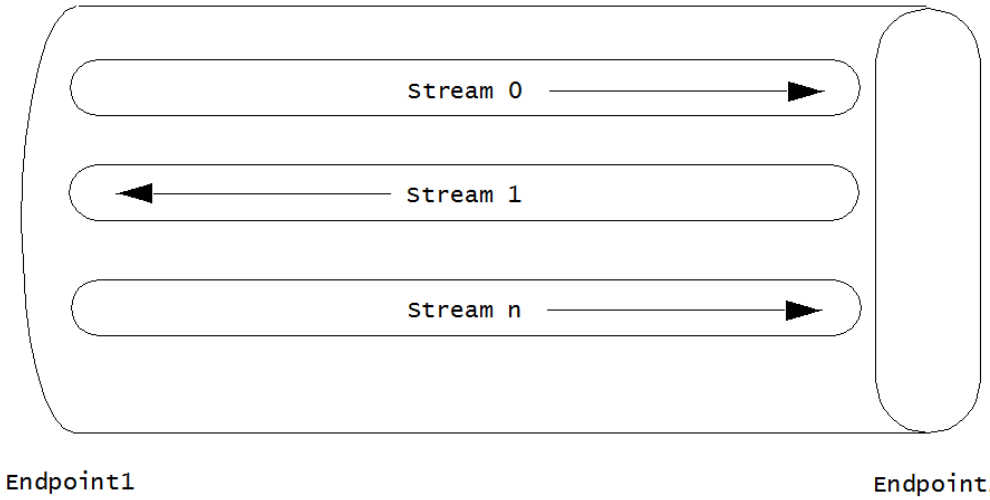


Congestion Control

SCTP congestion control mechanism is similar to that provided by TCP, and includes slow start, congestion avoidance, and fast retransmit. In SCTP, the initial congestion window (cwnd) is set to the double of the maximum transmission unit (MTU) while in TCP, it is usually set to one MTU. In SCTP, cwnd increases based on the number of acknowledged bytes, rather than the number of acknowledgements in TCP. The larger initial cwnd and the more aggressive cwnd adjustment provided by SCTP result in a larger average congestion window and, hence, better throughput performance than TCP.

Multi-Streaming

SCTP supports streams as depicted in the following figure which depicts an SCTP association that supports three streams.



The multiple stream mechanism is designed to solve the head-of-the-line blocking problem of TCP. Therefore, messages from different multiplexed flows do not block one another.

A stream can be thought of as a sub-layer between the transport layer and the upper layer. SCTP supports multiple logical streams to improve data transmission throughput. As shown in the above figure, SCTP allows multiple unidirectional streams within an association. This multiplexing/de-multiplexing capability is called multi-streaming and it is achieved by introducing a field called Stream Identifier contained in every DATA chunk) that is used to differentiate segments in different streams.

SIP transactions are mapped into SCTP streams as described in Section 5.1 of RFC 4168. In what it describes as the simplest way, the RFC suggests (keyword SHOULD) that all SIP messages be transmitted via Stream 0 with the U bit set to 1.

On the transmit side, the current SCTP implementation follows the RFC 4168 recommendation. On the receiving side, a SIP entity must be prepared to receive SIP messages over any stream.

Delivery Modes

SCTP supports two delivery modes, ordered and unordered . Delivery mode is specified by the U bit in the DATA chunk header — if the bit is clear (0), ordered delivery is specified; if the bit is set (1), unordered delivery is specified.

Within a stream, an SCTP endpoint must deliver ordered DATA chunks (received with the U bit set to 0) to the upper layer protocol according to the order of their Stream Sequence Number . Like the U bit, the Stream Sequence Number is a field within the DATA chunk header, and serves to identify the chunk's position with the message stream. If DATA chunks arrive out of order of their Stream Sequence Number, the endpoint must delay delivery to the upper layer protocol until they are reordered and complete.

Unordered DATA chunks (received with the U bit set to 1) are processed differently. When an SCTP endpoint receives an unordered DATA chunk, it must bypass the ordering mechanism and immediately deliver the data to the upper layer protocol (after reassembly if the user data is fragmented by the sender). As a consequence, the Stream Sequence Number field in an unordered DATA chunk has no significance. The sender can fill it with arbitrary value, but the receiver must ignore any value in field.

When an endpoint receives a DATA chunk with the U flag set to 1, it must bypass the ordering mechanism and immediately deliver the data to the upper layer (after reassembly if the user data is fragmented by the data sender).

Unordered delivery provides an effective way of transmitting out-of-band data in a given stream. Note also, a stream can be used as an unordered stream by simply setting the U bit to 1 in all DATA chunks sent through that stream.

Multi-Homing

Call control applications for carrier-grade service require highly reliable communication with no single point of failure. SCTP can assist carriers with its multi-homing capabilities. By providing different paths through the network over separate and diverse means, the goal of no single point of failure is more easily attained.

SCTP built-in support for multi-homed hosts allows a single SCTP association to run across multiple links or paths, hence achieving link/path redundancy. With this capability, and SCTP association can be made to achieve fast failover from one link/path to another with little interruption to the data transfer service.

Multi-homing enables an SCTP host to establish an association with another SCTP host over multiple interfaces identified by different IP addresses. With specific regard to the Oracle USM

these IP addresses need not be assigned to the same physical interface, or to the same physical Network Interface Unit.

If the SCTP nodes and the according IP network are configured in such a way that traffic from one node to another travels on physically different paths if different destination IP address are used, associations become tolerant against physical network failures and other problems of that kind.

An endpoint can choose an optimal or suitable path towards a multi-homed destination. This capability increases fault tolerance. When one of the paths fails, SCTP can still choose another path to replace the previous one. Data is always sent over the primary path if it is available. If the primary path becomes unreachable, data is migrated to a different, affiliated address — thus providing a level of fault tolerance. Network failures that render one interface of a server unavailable do not necessarily result in service loss. In order to achieve real fault resilient communication between two SCTP endpoints, the maximization of the diversity of the round-trip data paths between the two endpoints is encouraged.

Multi-Homing and Path Diversity

As previously explained, when a peer is multi-homed, SCTP can automatically switch the subsequent data transmission to an alternative address. However, using multi-homed endpoints with SCTP does not automatically guarantee resilient communications. One must also design the intervening network(s) properly.

To achieve fault resilient communication between two SCTP endpoints, one of the keys is to maximize the diversity of the round-trip data paths between the two endpoints. Under an ideal situation, one can make the assumption that every destination address of the peer will result in a different, separate path towards the peer. Whether this can be achieved in practice depends entirely on a combination of factors that include path diversity, multiple connectivity, and the routing protocols that glue the network together. In a normally designed network, the paths may not be diverse, but there may be multiple connectivity between two hosts so that a single link failure will not fail an association.

In an ideal arrangement, if the data transport to one of the destination addresses (which corresponds to one particular path) fails, the data sender can migrate the data traffic to other remaining destination address(es) (that is, other paths) within the SCTP association.

Monitoring Failure Detection and Recovery

When an SCTP association is established, a single destination address is selected as the primary destination address and all new data is sent to that primary address by default. This means that the behavior of a multi-homed SCTP association when there are no network losses is similar to behavior of a TCP connection. Alternate, or secondary, destination addresses are only used for redundancy purposes, either to retransmit lost packets or when the primary destination address cannot be reached.

A failover to an alternate destination is performed when the SCTP sender cannot elicit an acknowledgement — either a SACK for a DATA chunk, or a HEARTBEAT ACK for a HEARTBEAT chunk — for a configurable consecutive number of transmissions. The SCTP sender maintains an error-counter is maintained for each destination address and if this counter exceeds a threshold (normally six), the address is marked as inactive, and taken out of service. If the primary destination address is marked as inactive, all data is then switched to a secondary address to complete the failover.

If no data has been sent to an address for a specified time, that endpoint is considered to be idle and a HEARTBEAT packet is transmitted to it. The endpoint is expected to respond to the

HEARTBEAT immediately with a HEARTBEAT ACK. As well as monitoring the status of destination addresses, the HEARTBEAT is used to obtain RTT measurements on idle paths. The primary address becomes active again if it responds to a heartbeat.

The number of events where heartbeats were not acknowledged within a certain time, or retransmission events occurred is counted on a per association basis, and if a certain limit is exceeded, the peer endpoint is considered unreachable, and the association is closed.

The threshold for detecting an endpoint failure and the threshold for detecting a failure of a specific IP addresses of the endpoint are independent of each other. Each parameter can be separately configured by the SCTP user. Careless configuration of these protocol parameters can lead the association onto the dormant state in which all the destination addresses of the peer are found unreachable while the peer still remains in the reachable state. This is because the overall retransmission counter for the peer is still below the set threshold for detecting the peer failure.

Configuring SCTP Support for SIP

RFC 4168, *The Stream Control Transfer Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)*, specifies the requirements for SCTP usage as a layer 4 transport for SIP. Use the following steps to:

- configure SCTP as the layer 4 transport for a SIP interface
- create an SCTP-based SIP port
- associate physical interfaces/network interfaces with SIP realms
- identify adjacent SIP servers that are accessible via SCTP
- set SCTP timers and counters (optional)

Configuring an SCTP SIP Port

SIP ports are created as part of the SIP Interface configuration process.

1. From superuser mode, use the following command sequence to access sip-port configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)# sip-ports
   ORACLE(sip-port)#
   ```

2. Use the **address** parameter to provide the IPv4 or IPv6 address of the network interface that supports the SIP port.

   This is the primary address of a the local multi-homed SCTP endpoint.

   ```
   ORACLE(sip-port)# address 172.16.10.76
   ORACLE(sip-port)#
   ```

3. Retain the default value, 5060 (the well-known SIP port) for the **port** parameter.

   ```
   ORACLE(sip-port)# port 5060
   ORACLE(sip-port)#
   ```

4. Use the **transport-protocol** parameter to identify the layer 4 protocol.

   Supported values are UDP, TCP, TLS, and SCTP.

Select SCTP.

```
ORACLE(sip-port)# transport-protocol sctp
ORACLE(sip-port)#
```

5. Use the **multi-homed-addrs** parameter to specify one or more local secondary addresses of the SCTP endpoint.

   Multi-homed addresses must be of the same type (IPv4 or IPv6) as that specified by the **address** parameter. Like the address parameter, these addresses identify SD physical interfaces.

   To specify multiple addresses, bracket an address list with parentheses.

```
ORACLE(sip-port)# multi-homed-addrs 182.16.10.76
ORACLE(sip-port)#
ORACLE(sip-port)# multi-homed-addrs (182.16.10.76 192.16.10.76 196.15.32.108)
ORACLE(sip-port)#
```

6. Remaining parameters can be safely ignored.
7. Use **done**, **exit**, and **verify-config** to complete configuration of this SCTP-based SIP port.

```
ORACLE(sip-port)# done
ORACLE(sip-interface)# exit
ORACLE(session-router)# exit
ORACLE(configure)# exit
ORACLE# verify-config
----------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Configuring the Realm

After configuring a SIP port which identifies primary and secondary multi-homed transport addresses, you identify the network interfaces that support the primary address and secondary addresses to the realm assigned during SIP Interface configuration.

1. From superuser mode, use the following command sequence to access realm-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **select** command to access the target realm.
3. Use the **network-interfaces** command to identify the network interfaces that support the SCTP primary and secondary addresses.

   Network interfaces are identified by their name.

   Enter a list of network interface names using parentheses as list brackets. The order of interface names is not significant.

```
ORACLE(realm-config)# network-interfaces (mo1 M10)
ORACLE(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete realm configuration.

```
ORACLE(realm-config)# done
ORACLE(media-manager)# exit
ORACLE(configure)# exit
ORACLE# verify-config
```

```
---------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Configuring Session Agents

After configuring the realm, you identify adjacent SIP servers who will be accessed via the SCTP protocol.

1. From superuser mode, use the following command sequence to access session-agent configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

2. Use the **select** command to access the target session-agent.
3. Use the **transport-method** parameter to select the layer 4 transport protocol.

   Select staticSCTP for SCTP transport

   ```
   ORACLE(session-agent)# transport-method staticSCTP
   ORACLE(session-agent)#
   ```

4. Set the **reuse-connections** parameter to none.

   Select staticSCTP for SCTP transport

   ```
   ORACLE(session-agent)# reuse-connections none
   ORACLE(session-agent)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete session agent configuration.

   ```
   ORACLE(session-agent)# done
   ORACLE(session-router)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ---------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE#
   ```

6. Repeat Steps 1 through 5 as necessary to configure additional session agents who will be accessed via SCTP transport.

Setting SCTP Timers and Counters

Setting SCTP timers and counters is optional. All configurable timers and counters provide default values that conform to recommended values as specified in RFC 4960, Stream Control Transmission Protocol.

Management of Retransmission Timer, section 6.3 of RFC 4960 describes the calculation of a Retransmission Timeout (RTO) by the SCTP process. This calculation involves three SCTP protocol parameters: RTO.Initial, RTO.Min, and RTO.Max. Suggested SCTP Protocol Parameter Values section 15 of RFC 4960 lists recommended values for these parameters.

The following shows the equivalence of recommended values and ACLI defaults.

RTO.Initial = 3 seconds **sctp-rto-initial = 3000 ms (default value)**

RTO.Min = 1 second **sctp-rto-min = 1000 ms (default value)**

RTO.Max = 60 seconds **sctp-rto-max = 60000 ms (default value)**

Path Heartbeat, section 8.3 of RFC 4960 describes the calculation of a Heartbeat Interval by the SCTP process. This calculation involves the current calculated RTO and a single SCTP protocol parameter — HB.Interval.

The following shows the equivalence of recommended the value and ACLI default.

HB.Interval = 30 seconds **sctp-hb-interval = 3000 ms (default value)**

Acknowledgement on Reception of DATA Chunks, section 6.2 of RFC 4960 describes requirements for the timely processing and acknowledgement of DATA chunks. This section requires that received DATA chunks must be acknowledged within 500 milliseconds, and recommends that DATA chunks should be acknowledged with 200 milliseconds. The interval between DATA chunk reception and acknowledgement is specific by the ACLI **sctp-sack-timeout** parameter, which provides a default value of 200 milliseconds and a maximum value of 500 milliseconds.

Transmission of DATA Chunks, section 6.1 of RFC 4960 describes requirements for the transmission of DATA chunks. To avoid network congestion the RFC recommends a limitation on the volume of data transmitted at one time. The limitation is expressed in terms of DATA chunks, not in terms of SCTP packets.

The maximum number of DATA chunks that can be transmitted at one time is specified by the ACLI **sctp-max-burst** parameter, which provides a default value of 4 chunks, the limit recommended by the RFC.

Setting the RTO

An SCTP endpoint uses a retransmission timer to ensure data delivery in the absence of any feedback from its peer. RFC 4960 refers to the timer itself as T3-rtx and to the timer duration as RTO (retransmission timeout).

When an endpoint's peer is multi-homed, the endpoint calculates a separate RTO for each IP address affiliated with the peer. The calculation of RTO in SCTP is similar to the way TCP calculates its retransmission timer. RTO fluctuates over time in response to actual network conditions. To calculate the current RTO, an endpoint maintains two state variables per destination IP address — the SRTT (smoothed round-trip time) variable, and the RTTVAR (round-trip time variation) variable.

Use the following procedure to assign values used in RTO calculation.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

2. Use the **sctp-rto-initial** parameter to assign an initial timer duration.

   Allowable values are integers within the range 0 through 4294967295 that specify the initial duration in milliseconds. In the absence of an explicitly configured integer value, **sctp-rto-initial** defaults to 3000 milliseconds (3 seconds, the recommended default value from RFC 4960).

   As described in Section 6.3 of RFC 4960, the value specified by **sctp-rto-initial** is assigned to the SCTP protocol parameter RTO.Initial, which provides a default RTO until

actual calculations have derived a fluctuating duration based on network usage. The value specified by the **sctp-rto-initia**l parameter seeds these calculations.

```
ORACLE(network-parameters)# sctp-rto-initial 3000
ORACLE(network-parameters)#
```

3. Use the **sctp-rto-min** and **sctp-rto-max** parameters to assign an RTO floor and ceiling.

   Allowable values are integers within the range 0 through 4294967295 that specify the minimum and maximum durations in milliseconds. In the absence of an explicitly configured integer value, **sctp-rto-min** defaults to 1000 ms (1 second, the recommended default value from RFC 4960), and **sctp-rto-max** defaults to 60000 ms (60 seconds, the recommended default value from RFC 4960.)

   As described in Section 6.3 of RFC 4960, the values specified by **sctp-rto-min** and **sctp-rto-max** are assigned to the SCTP protocol parameters, RTO.min and RTO.max that limit RTO calculations. If a calculated RTO duration is less than RTO.min, the parameter value is used instead of the calculated value; likewise, if a calculated RTO duration is greater than RTO.max, the parameter value is used instead of the calculated value.

   ```
   ORACLE(network-parameters)# sctp-rto-min 1000
   ORACLE(network-parameters)# sctp-rto-max 60000
   ORACLE(network-parameters)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete RTO configuration.

   ```
   ORACLE(network-parameters)# done
   ORACLE(system)# exit
   ORACLE(configure)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE#
   ```

Setting the Heartbeat Interval

Both single-homed and multi-homed SCTP endpoints test the reachability of associates by sending periodic HEARTBEAT chunks to UNCONFIRMED or idle transport addresses.

Use the following procedure to assign values used in Heartbeat Interval calculation.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

2. Use the **sctp-hb-interval** parameter to assign an initial Heartbeat Interval duration.

   Allowable values are integers within the range 0 through 4294967295 that specify the initial Heartbeat Interval in milliseconds. In the absence of an explicitly configured integer value, **sctp-hb-interval** defaults to 30000 milliseconds (30 seconds, the recommended default value from RFC 4960).

   As described in Section 8.3 of RFC 4960, the value specified by **sctp-hb-interval** is assigned to the SCTP protocol parameter HB.Interval, which provides a default interval until actual calculations have derived a fluctuating interval based on network usage. The value specified by the **sctp-hb-interval** parameter is used during these calculations.

```
ORACLE(network-parameters)# sctp-hb-interval 30000
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete Heartbeat Interval configuration.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE #
```

Setting the SACK Delay Timer

An SCTP Selective Acknowledgement (SACK) is sent to the peer endpoint to acknowledge received DATA chunks and to inform the peer endpoint of gaps in the received subsequences of DATA chunks. Section 6.2 of RFC 4960 sets a specific requirement for a SACK Delay timer that specifies the maximum interval between the reception of an SCTP packet containing one or more DATA chunks and the transmission of a SACK to the packet originator.

Use the following procedure to set the SACK Delay timer.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

2. Use the **sctp-sack-timeout** parameter to assign a value to the SACK Delay timer.

   Allowable values are integers within the range 0 through 500 which specify the maximum delay (in milliseconds) between reception of a SCTP packet containing one or more Data chunks and the transmission of a SACK to the packet source. The value 0 indicates that a SACK is generated immediately upon DATA chunk reception

   In the absence of an explicitly configured integer value, **sctp-sack-timeout** defaults to 200 ms (the recommended default value from RFC 4960).

```
ORACLE(network-parameters)# sctp-sack-timeout 200
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete configuration of the SACK Delay timer.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Limiting DATA Bursts

Section 6.1 of RFC 4960 describes the SCTP protocol parameter, Max.Burst, used to limit the number of DATA chunks that are transmitted at one time.

Use the following procedure to assign a value to the SCTP protocol parameter, Max.Burst.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

2. Use the **sctp-max-burst** parameter to assign a value to the SCTP protocol parameter, Max.Burst.

   Allowable values are integers within the range 0 through 4294967295 that specify the maximum number of DATA chunks that will be sent at one time. In the absence of an explicitly configured integer value, **sctp-max-burst** defaults to 4 (DATA chunks, the recommended default value from RFC 4960).

```
ORACLE(network-parameters)# sctp-max-burst 4
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete configuration of DATA burst limitations.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
-------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Setting Endpoint Failure Detection

As described in Monitoring, Failure Detection and Recovery, a single-homed SCTP endpoint maintains a count of the total number of consecutive failed (unacknowledged) retransmissions to its peer. Likewise, a multi-homed SCTP endpoint maintains a series of similar, dedicated counts for all of its destination transport addresses. If the value of these counts exceeds the limit indicated by the SCTP protocol parameter Association.Max.Retrans, the endpoint considers the peer unreachable and stops transmitting any additional data to it, causing the association to enter the CLOSED state.

The endpoint resets the counter when (1) a DATA chunk sent to that peer endpoint is acknowledged by a SACK, or (2) a HEARTBEAT ACK is received from the peer endpoint.

Use the following procedure to configure endpoint failure detection.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

2. Use the **sctp-assoc-max-retrans** to assign a value to the SCTP protocol parameter Association.Max.Retrans.

   Allowable values are integers within the range 0 through 4294967295 which specify the maximum number of transmission requests. In the absence of an explicitly configured integer value, s**ctp-assoc-max-retrans** defaults to 10 (transmission re-tries, the recommended default value from RFC 4960).

```
ORACLE(network-parameters)# sctp-assoc-max-retrans 10
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete endpoint failure detection configuration.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
----------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Setting Path Failure Detection

As described in Monitoring, Failure Detection and Recovery, when its peer endpoint is multi-homed, an SCTP endpoint maintains a count for each of the peer's destination transport addresses.

Each time the T3-rtx timer expires on any address, or when a HEARTBEAT sent to an idle address is not acknowledged within an RTO, the count for that specific address is incremented. If the value of a specific address count exceeds the SCTP protocol parameter Path.Max.Retrans, the endpoint marks that destination transport address as inactive.

The endpoint resets the counter when (1) a DATA chunk sent to that peer endpoint is acknowledged by a SACK, or (2) a HEARTBEAT ACK is received from the peer endpoint.

When the primary path is marked inactive (due to excessive retransmissions, for instance), the sender can automatically transmit new packets to an alternate destination address if one exists and is active. If more than one alternate address is active when the primary path is marked inactive, a single transport address is chosen and used as the new destination transport address.

Use the following procedure to configure path failure detection.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-parameters
ORACLE(network-parameters)#
```

2. Use the **sctp-path-max-retrans** parameter to assign a value to the SCTP protocol parameter Path.Max.Retrans.

   Allowable values are integers within the range 0 through 4294967295 that specify the maximum number of RTOs and unacknowledged HEARTBEATS. In the absence of an explicitly configured integer value, **sctp-path-max-retrans** defaults to 5 (RTO and/or HEARTBEAT errors per transport address, the recommended default value from RFC 4960).

   When configuring endpoint and path failure detection, ensure that the value of the **sctp-assoc-max-retrans** parameter is smaller than the sum of the **sctp-path-max-retrans** values for all the remote peer's destination addresses. Otherwise, all the destination addresses can become inactive (unable to receive traffic) while the endpoint still considers the peer endpoint reachable.

```
ORACLE(network-parameters)# sctp-path-max-retrans 5
ORACLE(network-parameters)#
```

3. Use **done**, **exit**, and **verify-config** to complete path failure detection configuration.

```
ORACLE(network-parameters)# done
ORACLE(system)# exit
ORACLE(configure)# exit
ORACLE(configure)# exit
ORACLE# verify-config
----------------------------------------------------------------------
Verification successful! No errors nor warnings in the configuration
ORACLE#
```

Specifying the Delivery Mode

As described in Delivery Modes, SCTP support two delivery modes, ordered and unordered.

1. From superuser mode, use the following command sequence to access network-parameters configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# network-parameters
   ORACLE(network-parameters)#
   ```

2. Use the **sctp-send-mode** parameter to select the preferred delivery mode.

   Choose ordered or unordered.

   ```
   ORACLE(network-parameters)# sctp-send-mode unordered
   ORACLE(network-parameters)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete delivery mode configuration.

   ```
   ORACLE(network-parameters)# done
   ORACLE(system)# exit
   ORACLE(configure)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE #
   ```

Example Configurations

The following ACLI command sequences summarize required SCTP port configuration, and the configuration of required supporting elements.

- PHY interfaces
- Network interfaces
- SIP ports
- realms
- session agents

Sequences show only configuration parameters essential for SCTP operations; other parameters can retain default values, or assigned other values specific to local network requirements.

Phy Interface Configuration

The first ACLI command sequence configures a physical interface named m10, that will support an SCTP primary address; the second sequence configures an interface named m01 that will support a secondary SCTP address.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# phy-interface
ORACLE(phy-interface)# operation-type media
ORACLE(phy-interface)# port 0
ORACLE(phy-interface)# slot 1
ORACLE(phy-interface)# name m10
ORACLE(phy-interface)#
...
...
...
ORACLE(phy-interface)#
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# phy-interface
ORACLE(phy-interface)# operation-type media
ORACLE(phy-interface)# port 1
ORACLE(phy-interface)# slot 0
ORACLE(phy-interface)# name m01
ORACLE(phy-interface)#
...
...
...
ORACLE(phy-interface)#
```

Network Interface Configuration

These ACLI command sequences configure two network interfaces. The first sequence configures a network interface named m10, thus associating the network interface with the physical interface of the same name. The ACLI **ip-address** command assigns the IPv4 address 172.16.10.76 to the network interface. In a similar fashion, the second command sequence associates the m01 network and physical interfaces, and assigns an IPv4 address of 182.16.10.76.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)# name m10
ORACLE(network-interface)# ip-address 172.16.10.76
...
...
...
ORACLE(network-interface)#
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# network-interface
ORACLE(network-interface)# name m01
ORACLE(network-interface)# ip-address 182.16.10.76
...
...
...
ORACLE(network-interface)#
```

SIP Port Configuration

This ACLI command sequence configures a SIP port for SCTP operations. It specifies the use of SCTP as the transport layer protocol, and assigns the existing network interface address, 172.16.10.76, as the SCTP primary address. Additionally, it identifies three other existing

network addresses (182.16.10.76, 192.16.10.76, and 196.15.32.108) as SCTP secondary addresses.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# sip-ports
ORACLE(sip-port)# address 172.16.10.76
ORACLE(sip-port)# transport-protocol sctp
ORACLE(sip-port)# multi-homed-addrs (182.16.10.76 192.16.10.76 196.15.32.108)
...
...
...
ORACLE(sip-port)#
```

Realm Configuration

These ACLI command sequences configure a realm for SCTP operations. The first ACLI sequence assigns a named realm, in this example core-172, to a SIP interface during the interface configuration process. The second sequence accesses the target realm and uses the **network-interfaces** command to associate the named SCTP network interfaces with the realm.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# realm-id core-172
...
...
...
ORACLE(sip-interface)#
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier: core-172
1. core-172 ...
selection: 1
ORACLE(realm-config)# network-interfaces (m01 m10 ...)
...
...
...
ORACLE(ream-config)#
```

Session Agent Configuration

The final ACLI command sequence enables an SCTP-based transport connection between the Acme Packet 4500 and an adjacent network element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)# select
<hostname>: core-172S1
1. core-172S1 ...
selection: 1
ORACLE(session-agent)#
ORACLE(session-agent)# transport-method staticSCTP
ORACLE(session-agent)# reuse-connections none
...
...
```

```
...
ORACLE(session-agent)#
```

IPV6 Address Configuration

IPv6 can be a licensed feature on the Oracle USM. IPv6 is available on the Oracle CSM without an explicit license or entitlement.

If you want to add this license to a system, contact Oracle. Once you have the license, refer to the Getting Started chapter for instructions about how to add a license.

You do not need to take action if you are working with a new system with which the IPv6 license was purchased.

> **Note:**
>
> For ACLI parameters that support only IPv4, there are many references to that version as the accepted value for a configuration parameter or other IPv4-specific languages. For IPv6 support, these references have been edited. For example, rather than providing help that refers specifically to IPv4 addresses when explaining what values are accepted in an ACLI configuration parameter, you will now see an <ipAddr> note.

This section calls out the configurations and parameters for which you can enter IPv6 addresses. In this first IPv6 implementation, the complete range of system configurations and their parameters are available for IPv6 use.

The Oracle USM follows RFC 3513 its definition of IPv6 address representations. Quoting from that RFC, these are the two forms supported:

- The preferred form is x:x:x:x:x:x:x:x, where the 'x's are the hexadecimal values of the eight 16-bit pieces of the address. Examples:

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

1080:0:0:0:8:800:200C:417A

Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.).

- Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address. For example, the following addresses: 1080:0:0:0:8:800:200C:417A a unicast address FF01:0:0:0:0:0:0:101 a multicast address
  0:0:0:0:0:0:0:1 the loopback address

  0:0:0:0:0:0:0:0 the unspecified addresses

  may be represented as:

  1080::8:800:200C:417A a unicast address

  FF01::101 a multicast address

  ::1 the loopback address

:: the unspecified addresses

Access Control

These are the IPv6-enabled parameters in the **access-control** configuration.

Parameter	Entry Format
source-address	<ip-address>[/<num-bits>][:<port>[/<port-bits>]]
destination-address	<ip-address>[/<num-bits>][:<port>[/<port-bits>]]

Host Route

These are the IPv6-enabled parameters in the **host-route** configuration.

Parameter	Entry Format
dest-network	<ipv4> | <ipv6>
netmask	<ipv4> | <ipv6>
gateway	<ipv4> | <ipv6>

Local Policy

These are the IPv6-enabled parameters in the **local-policy** configuration.

Parameter	Entry Format
from-address	<ipv4> | <ipv6> | POTS Number, E.164 Number, hostname, wildcard
to-address	<ipv4> | <ipv6> | POTS Number, E.164 Number, hostname, wildcard

Network Interface

These are the IPv6-enabled parameters in the **network-interface** configuration.

Parameter	Entry Format
hostname	<ipv4> | <ipv6> | hostname
ip-address	<ipv4> | <ipv6>
pri-utility-addr	<ipv4> | <ipv6>
sec-utility-addr	<ipv4> | <ipv6>
netmask	<ipv4> | <ipv6>
gateway	<ipv4> | <ipv6>
sec-gateway	<ipv4> | <ipv6>
dns-ip-primary	<ipv4> | <ipv6>
dns-ip-backup1	<ipv4> | <ipv6>
dns-ip-backup2	<ipv4> | <ipv6>
add-hip-ip	<ipv4> | <ipv6>
remove-hip-ip	<ipv4> | <ipv6>
add-icmp-ip	<ipv4> | <ipv6>
remove-icmp-ip	<ipv4> | <ipv6>

ENUM Server

These are the IPv6-enabled parameters in the enum-config.

Parameter	Entry Format
enum-servers	[<ipv4> | <ipv6>]:port

Realm Configuration

These are the IPv6-enabled parameters in the **realm-config**.

Parameter	Entry Format
addr-prefix	[<ipv4> | <ipv6>]/prefix

Session Agent

These are the IPv6-enabled parameters in the **session-agent** configuration.

Parameter	Entry Format
hostname	<ipv4> | <ipv6>
ip-address	<ipv4> | <ipv6>

SIP Configuration

These are the IPv6-enabled parameters in the **session-config**.

Parameter	Entry Format
registrar-host	<ipv4> | <ipv6> | hostname | *

SIP Interface SIP Ports

These are the IPv6-enabled parameters in the **sip-interface>sip-ports** configuration.

Parameter	Entry Format
address	<ipv4> | <ipv6>

Steering Pool

These are the IPv6-enabled parameters in the **steering-pool** configuration.

Parameter	Entry Format
ip-address	<ipv4> | <ipv6>

System Configuration

These are the IPv6-enabled parameters in the **system-config**.

Parameter	Entry Format
default-v6-gateway	<ipv6>

IPv6 Support for Management and Telemetry

Several management-oriented parameters on the Oracle USM may be configured with IPv6 addresses to be used within IPv6 networks.

The following parameters that are configured with IP addresses accept IPv6 addresses to be used within IPv6 address space.

You may configure the wancom0/eth0 physical interface in the bootparams with an IPv6 address and complementary IPv6 gateway via the following parameters:

- **bootparams** > **inet on ethernet**
- **bootparams** > **gateway inet**

You may configure a syslog server with an IPv6 destination address via the following parameter:

- **system** > **system-config** > **syslog-servers** > **address**

You may configure a system access list entry with an IPv6 source address and complementary IPv6 gateway.

- **system** > **system-access-list** > **source-address**
- **system** > **system-access-list** > **netmask**

You may configure a RADIUS server with an IPv6 destination address via the following parameter:

- **security** > **authentication** > **radius-servers** > **address**

IPv6 Default Gateway

In the system configuration, you configure a default gateway—a parameter that now has its own IPv6 equivalent.

To configure an IPv6 default gateway:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **system-config** and press Enter.

```
ORACLE(system)# system-config
ORACLE(system-config)#
```

4. **default-v6-gateway**—Set the IPv6 default gateway for this Oracle USM. This is the IPv6 egress gateway for traffic without an explicit destination. The application of your Oracle USM determines the configuration of this parameter.
5. Save your work.

IPv6 Link Local Addresses

The Oracle USM supports IPv6 Link Local addresses configured for a network interface's gateway.

An IPv6 link local address is signified by its first hextet set to FE80:. Even if a network interface's first hextet is not FE80, but the gateway is, the Oracle USM will still function as expected.

show neighbor-table

The show neighbor-table command displays the IPv6 neighbor table and validates that there is an entry for the link local address, and the gateway uses that MAC address.

```
System# show neighbor-table
LINK LEVEL NEIGHBOR TABLE
Neighbor                              Linklayer Address  Netif Expire    S Flags
300::100                              0:8:25:a1:ab:43      sp0 permanent ? R
871962224
400::100                              0:8:25:a1:ab:45      sp1 permanent ? R
871962516
fe80::bc02:a98f:f61e:20%sp0           be:2:ac:1e:0:20      sp0 4s        ? R
871962808
fe80::bc01:a98f:f61e:20%sp1           be:1:ac:1e:0:20      sp1 4s        ? R
871963100
------------------------------------------------------------------------------
ICMPv6 Neighbor Table:
------------------------------------------------------------------------------
---------
  entry: slot port vlan IP                              type       flag pendBlk
Hit MAC
------------------------------------------------------------------------------
---------
  5    : 1    0    0    fe80::bc01:a98f:f61e:20/64     08-DYNAMIC 1    0
1   be:01:ac:1e:00:20
  4    : 1    0    0    0.0.0.0/64                     01-GATEWAY 0    0
1   be:01:ac:1e:00:20
  3    : 1    0    0    400::/64                       02-NETWORK 0    0
1   00:00:00:00:00:00
  2    : 0    0    0    fe80::bc02:a98f:f61e:20/64     08-DYNAMIC 1    0
1   be:02:ac:1e:00:20
  1    : 0    0    0    0.0.0.0/64                     01-GATEWAY 0    0
1   be:02:ac:1e:00:20
  0    : 0    0    0    300::/64                       02-NETWORK 0    0
1   00:00:00:00:00:00
------------------------------------------------------------------------------
---------
```

Network Interfaces and IPv6

You set many IP addresses in the network interface, one of which is the specific IP address for that network interface and others that are related to different types of management traffic. This section outlines rules you must follow for these entries.

- For the **network-interface ip-address** parameter, you can set a single IP address. When you are working with an IPv6-enabled system, however, note that all other addresses related to that network-interface IP address must be of the same version.
- Heterogeneous address family configuration is prevented for the **dns-ip-primary**, **dns-ip-backup1**, and **dns-ip-backup2** parameters.
- For HIP addresses (**add-hip-ip**), you can use either IPv4 or IPv6 entries.
- For ICMP addresses (**add-icmp-ip**), you can use either IPv4 or IPv6 entries.
- For Telnet (**add-telnet-ip**), FTP (**add-ftp-ip**), and SNMP (**add-snmp-ip**), you are not allowed to use IPv6; your entries MUST use IPv4.

IPv6 Reassembly and Fragmentation Support

As it does for IPv4, the Oracle USM supports reassembly and fragmentation for large signaling packets when you enable IPV6 on your system.

The Oracle USM takes incoming fragments and stores them until it receives the first fragment containing a Layer 4 header. With that header information, the Oracle USM performs a look-up so it can forward the packets to its application layer. Then the packets are re-assembled at the applications layer. Media fragments, however, are not reassembled and are instead forwarded to the egress interface.

On the egress side, the Oracle USM takes large signaling messages and encodes it into fragment datagrams before it transmits them.

Note that large SIP INVITE messages should be sent over TCP. If you want to modify that behavior, you can use the SIP interface's option parameter **max-udp-length=xx** for each SIP interface where you expect to receive large INVITE packets.

Other than enabling IPv6 on your Oracle USM, there is no configuration for IPv6 reassembly and fragmentation support. It is enabled automatically.

Access Control List Support

The Oracle USM supports IPv6 for access control lists in two ways:

- For static access control lists that you configure in the **access-control** configuration, your entries can follow IPv6 form. Further, this configuration supports a prefix that enables wildcarding the source IP address.
- Dynamic ACLs are also supported; the Oracle USM will create ACLs for offending IPv6 endpoints.

Data Entry

When you set the **source-address** and **destination-address** parameters in the **access-control** configuration, you will use a slightly different format for IPv6 than for IPv4.

For the **source-address**, your IPv4 entry takes the following format: <ip-address>[/<num-bits>][:<port>[/<port-bits>]]. And for the **destination-address**, your IPv4 entry takes this format: <ip-address>[:<port>[/<port-bits>]].

Since the colon (:) in the IPv4 format leads to ambiguity in IPv6, your IPv6 entries for these settings must have the address encased in brackets ([]): [7777::11]/64:5000/14.

In addition, IPv6 entries are allowed up to 128 bits for their prefix lengths.

The following is an example access control configuration set up with IPv6 addresses.

```
ORACLE(access-control)# done
access-control
        realm-id                        net7777
        description
        source-address                  7777::11/64:5060/8
        destination-address             8888::11:5060/8
        application-protocol            SIP
        transport-protocol              ALL
        access                          deny
        average-rate-limit              0
        trust-level                     none
        minimum-reserved-bandwidth      0
        invalid-signal-threshold        10
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     30
```

DNS Support

The Oracle USM supports the DNS resolution of IPv6 addresses; in other words, it can request the AAAA record type (per RFC 1886) in DNS requests. In addition, the Oracle USM can make DNS requests over IPv6 transport so that it can operate in networks that host IPv6 DNS servers.

For mixed IPv4-IPv6 networks, the Oracle USM follows these rules:

- If the realm associated with the name resolution is an IPv6 realm, the Oracle USM will send the query out using the AAAA record type.
- If the realm associated with the name resolution is an IPv4 realm, the Oracle USM will send the query out using the A record type.

In addition, heterogeneous address family configuration is prevented for the **dns-ip-primary**, **dns-ip-backup1**, and **dns-ip-backup2** parameters.

Homogeneous Realms

IPv6 is supported for realms and for nested realms, as long as the parent chain remains within the same address family. If you try to configure realms with mixed IPv4-IPv6 addressing, your system will issue an error message when you try to save your configuration. This check saves you time because you do not have to wait to run a configuration verification (using the ACLI **verify-config** command) to find possible errors.

Parent-Child Network Interface Mismatch

Your system will issue the following error message if parent-child realms are on different network interfaces that belong to different address families:

```
ERROR: realm-config [child] and parent [net8888] are on network interfaces that
belong to different address families
```

Address Prefix-Network Interface Mismatch

If the address family and the address-prefix you configure for the realm does not match the address family of its network interface, your system will issue the following error message:

```
ERROR: realm-config [child] address prefix and network interface [1:1:0] belong
to different address families
```

RADIUS Support for IPv6

The Oracle USM's RADIUS support now includes:

- RADIUS CDR generation for SIPv6-SIPv6 and SIPv6-SIPv4 calls
- IPv6-based addresses in RADIUS CDR attributes

This means that for the CDR attributes in existence prior to the introduction of IPv6 to the Acme Packet 4500 are mapped to the type ipaddr, which indicates four-byte field. The sixteen-byte requirement for IPv6 addresses is now supported, and there are a parallel set of attributes with the type ipv6addr. Attributes 155-170 are reserved for the IPv6 addresses.

NAS addresses use the number 95 to specify the NAS-IPV6-Address attribute. And local CDRs now contain IPv6 addresses.

Supporting RADIUS VSAs

The following VSAs have been added to the Oracle RADIUS dictionary to support IPv6.

```
Acme-Flow-In-Src-IPv6_Addr_FS1_F        155     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS1_F        156     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS1_F       157     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS1_F       158     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS1_R        159     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS1_R        160     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS1_R       161     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS1_R       162     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS2_F        163     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS2_F        164     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS2_F       165     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS2_F       166     ipv6addr      Acme
Acme-Flow-In-Src-IPv6_Addr_FS2_R        167     ipv6addr      Acme
Acme-Flow-In-Dst-IPv6_Addr_FS2_R        168     ipv6addr      Acme
Acme-Flow-Out-Src-IPv6_Addr_FS2_R       169     ipv6addr      Acme
Acme-Flow-Out-Dst-IPv6_Addr_FS2_R       170     ipv6addr      Acme
```

4

Realms and Nested Realms

This chapter explains how to configure realms and nested realms, and specialized media-related features.

A realm is a logical definition of a network or groups of networks made up in part by devices that provide real-time communication sessions comprised of signaling messages and possibly media flows. These network devices might be call agents, softswitches, SIP proxies, IP PBXs, etc., that are statically defined by IPv4 addresses. These network devices might also be IPv4 endpoints: SIP phones, IADs, MAs, media gateways, etc., that are defined by an IPv4 address prefix.

Realms support bandwidth-based call admission control and QoS marking for media. They are the basis for defining egress and ingress traffic to the Oracle USM—which supports the Oracle USM's topology hiding capabilities.

This chapter also explains how to configure media ports (steering pools). A steering pool exists within a realm and contains a range of ports that have a common address (for example, a target IPv4 address). The range of ports contained in the steering pool are used to steer media flows from one realm, through the Oracle USM, to another.

Finally, in this chapter you can learn about TOS/DiffServ functionality for realm-based packet marking by media type.

Overview

Realms are a logical distinction representing routes (or groups of routes) reachable by the Oracle USM and what kinds of resources and special functions apply to those routes. Realms are used as a basis for determining ingress and egress associations to network interfaces, which can reside in different VPNs. The ingress realm is determined by the signaling interface on which traffic arrives. The egress realm is determined by the following:

- Routing policy—Where the egress realm is determined in the session agent configuration or external address of a SIP-NAT
- Realm-bridging—As applied in the SIP-NAT configuration configurations
- Third-party routing/redirect (i.e., SIP redirect)

Realms also provide configuration support for denial of service (DoS)/access control list (ACL) functionality.

Realms can also be nested in order to form nested realm groups. Nested realms consist of separate realms that are arranged within a hierarchy to support network architectures that have separate backbone networks and VPNs for signaling and media. This chapter provides detailed information about nested realms after showing you how to configure realms on your Oracle USM.

About Realms and Network Interfaces

All realms reference network interfaces on the Oracle USM. This reference is made when you configure a list of network interfaces in the realm configuration.

You configure a network interface to specify logical network interfaces that correspond existing physical interfaces on the Oracle USM. Configuring multiple network interfaces on a single physical interface creates a channelized physical interface, a VLAN. VLANs, in turn, allow you to reuse address space, segment traffic, and maximize bandwidth.

In order to reach the realms you configure, you need to assign them network interfaces. The values you set for the name and port in the network interface you select then indicate where the realm can be reached.

About the SIP Home Realm

The realm configuration is also used to establish what is referred to as the SIP home realm. This is the realm where the Oracle USM's SIP proxy sits.

In peering configurations, the SIP home realm is the internal network of the SIP proxy. In backbone access configurations, the SIP home realm typically interfaces with the backbone connected network. In additions, the SIP home realm is usually exposed to the Internet in an HNT configuration.

Although you configure a SIP home realm in the realm configuration, it is specified as the home realm in the main SIP configuration by the home realm identifier parameter. Specifying the SIP home realm means that the Oracle USM's SIP proxy can be addressed directly by connected entities, but other connected network signaling receives layer 3 NAT treatment before reaching the internal SIP proxy.

About Realms and Other Oracle USM Functions

Realms are referenced by other configurations in order to support this functionality across the protocols the Oracle USM supports and to make routing decisions. Other configurations' parameters that point to realms are:

- SIP configuration: home realm identifier, egress realm identifier
- SIP-NAT configuration: realm identifier
- Session agent configuration: realm identifier
- Media manager: home realm identifier
- Steering ports: realm identifier
- Static flow: in realm identifier, out realm identifier

Realms

Realm configuration is divided into the following functional areas, and the steps for configuring each are set out in this chapter: identity and IP address prefix, realm interfaces, realm service profiles, QoS measurement, QoS marking, address translation profiles, and DNS server configuration.

Before You Configure

Before you configure realms, you want to establish the physical and network interfaces with which the realm will be associated.

- Configure a physical interface to define the physical characteristics of the signaling line.

- Configure a network interface to define the network in which this realm is participating and optionally to create VLANs.

If you wish to use QoS, you should also determine if your Oracle USM is QoS enabled.

Remember that you will also use this realm in other configurations to accomplish the following:

- Set a signaling port or ports at which the Oracle USM listens for signaling messages.
- Configure sessions agents to point to ingress and egress signaling devices located in this realm in order to apply constraint for admission control.
- Configure session agents for defining trusted sources for accepting signaling messages.

Realm Configuration

To access the realm configuration parameters in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

Identity and IP Address Prefix

The first parameters you configure for a realm are its name (a unique identifier) and an IP address prefix and subnet mask.

The IP address and subnet mask establish a set of matching criteria for the realm, and distinguishes between realms that you assign to the same network interface.

To configure a realm's identity and IP address prefix in the ACLI:

1. **identifier**—Enter the name of the realm. This parameter uniquely identifies the realm. You will use this parameter in other configurations when asked for a realm identifier value.
2. **addr-prefix**—Enter the IPv4 or IPv6 address and subnet mask combination to set the criteria the Oracle USM uses to match packets sent or received on the network interface associated with this realm. This matching determines the realm, and subsequently what resources are used for that traffic. This setting determines whether the realm is an IPv4 or IPv6 realm.

   This parameter must be entered in the correct format where the IPv4 or IPv6 address comes first and is separated by a slash (/) from the subnet mask value. For example, 172.16.0.0/24.

   The default for this parameter is **0.0.0.0**. When you leave this parameter set to the default, all addresses match.

Realm Interfaces

The realm points to one network interface on the Oracle USM.

> **Note:**
>
> Only one network-interface can be assigned to a single realm-config object, except for Local multi-homing SCTP deployments.

To assign interfaces to a realm:

1. **network-interfaces**—Enter the physical and network interface(s) that you want this realm to reference. These are the network interfaces though which this realm can be reached by ingress traffic, and through which this traffic exits the system as egress traffic.

   Enter the name and port in the correct format where the name of the interface comes first and is separated by a colon (:) from the port number. For example, f10:0.

   The parameters you set for the network interfaces must be unique.

   Enter multiple network interfaces for this list by typing an open parenthesis, entering each field value separated by a Space, typing a closed parenthesis, and then pressing Enter.

   ```
   ORACLE(realm-config)# network-interfaces fe1:0
   ```

   You must explicitly configure a realm's network interface as either IPv4 or IPv6 when the applicable interface is either dual-stack or IPv6. You do this by appending the realm's network-interface with a **.4** or a **.6**, as shown below.

   ```
   ORACLE(realm-config)# network-interfaces fe1:0.6
   ```

   For single-stack interface configurations that do not specify this format, the Oracle USM assumes an IPv4 interface. Dual stack interface configurations fail if this IP version family suffix is not specified.

Realm Service Profile

The parameters you configure to establish the realm service profile determine how bandwidth resources are used and how media is treated in relation to the realm. Bandwidth constraints set for realm service profiles support the Oracle USM's admission control feature.

Peer-to-peer media between endpoints can be treated in one of three different ways:

- Media can be directed between sources and destinations within this realm on this specific Oracle USM. Media travels through the Oracle USM rather than straight between the endpoints.
- Media can be directed through the Oracle USM between endpoints that are in different realms, but share the same subnet.
- For SIP only, media can be released between multiple Oracle USMs.
  To enable SIP distributed media release, you must set the appropriate parameter in the realm configuration. You must also set the SIP options parameter to media-release with the appropriate header name and header parameter information. This option defines how the Oracle USM encodes IPv4 address and port information for media streams described by, for example, SDP.

To configure realm service profile:

1. **max-bandwidth**—Enter the total bandwidth budget in kilobits per second for all flows to/from the realm defined in this element. The default is **0** which allows for unlimited bandwidth. The valid range is:
   - Minimum—0
   - Maximum—4294967295
2. **mm-in-realm**—Enable this parameter to treat media within this realm on this Oracle USM. The default is **disabled**. Valid values are:
   - enabled | disabled
3. **mm-in-network**—Enable this parameter to treat media within realms that have the same subnet mask on this Oracle USM. The default is **enabled**. Valid values are:
   - enabled | disabled
4. **msm-release**—Enable or disable the inclusion of multi-system (multiple Oracle USMs) media release information in the SIP signaling request sent into the realm identified by this realm-config element. If this field is set to enabled, another Oracle USM is allowed to decode the encoded SIP signaling request message data sent from a SIP endpoint to another SIP endpoint in the same network to resore the original SDP and subsequently allow the media to flow directly between those two SIP endpoints in the same network serviced by multiple Oracle USMs. If this field is disabled, the media and signaling will pass through boht Oracle USMs. Remember that for this feature to work, you must also set the options parameter in the SIP configuration accordingly. The default is **disabled**. Valid values are:
   - enabled | disabled

QoS Measurement

This chapter provides detailed information about when to configure the **qos-enable** parameter. If you are not using QoS or a QoS-capable Oracle USM, then you can leave this parameter set to disabled (default).

QoS Marking

QoS marking allows you to apply a set of TOS/DiffServ mechanisms that enable you to provide better service for selected networks

You can configure a realm to perform realm-based packet marking by media type, either audio/voice or video.

The realm configuration references a set of media policies that you configure in the media policy configuration. Within these policies, you can establish TOS/DiffServ values that define an individual type (or class) of service, and then apply them on a per-realm basis. In the media profiles, you can also specify:

- One or more audio media types for SIP
- One or more video types for SIP
- Both audio and video media types for SIP
  To establish what media policies to use per realm in the ACLI:

1. **media-policy**—Enter the name (unique identifier) of the media policy you want to apply in the realm. When the Oracle USM first sets up a SIP media session, it identifies the egress

realm of each flow and then determines the media-policy element to apply to the flow. This parameter must correspond to a valid name entry in a media policy element. If you leave this parameter empty, then QoS marking for media will not be performed for this realm.

Address Translation Profiles

If you are not using this feature, you can leave the **in-translationid** and **out-translationid** parameters blank.

DNS Servers

You can configure DNS functionality on a per-network-interface basis, or you can configure DNS servers to use per realm. Configuring DNS servers for your realms means that you can have multiple DNS servers in connected networks. In addition, this allows you to specify which DNS server to use for a given realm such that the DNS might actually be in a different realm with a different network interface.

This feature is available for SIP only.

To configure realm-specific DNS in the ACLI:

1. **dns-realm**—Enter the name of the network interface that is configured for the DNS service you want to apply in this realm. If you do not configure this parameter, then the realm will use the DNS information configured in its associated network interface.

DoS ACL Configuration

If you are not using this functionality, you can leave the parameters at their default values: **average-rate-limit**, **peak-rate-limit**, **maximum-burst-size**, **access-control-trust-level**, **invalid-signal-threshold**, and **maximum-signal-threshold**.

Enabling RTP-RTCP UDP Checksum Generation

You can configure the Oracle USM to generate a UDP checksum for RTP/ RTCP packets on a per-realm basis. This feature is useful in cases where devices performing network address translation (NAT) do not pass through packets with a zero checksum from the public Internet. These packets do not make it through the NAT, even if they have the correct to and from IP address and UDP port information. When you enable this feature, the Oracle USM calculates a checksum for these packets and thereby enables them to traverse a NAT successfully.

If you do not enable this feature, then the Oracle USM will not generate a checksum for RTP or RTCP packets if their originator did not include one. If a checksum is already present when the traffic arrives at the hardware, the system will relay it.

You enable this feature on the outbound realm.

Aggregate Session Constraints Per Realm

You can set session constraints for the Oracle USM's global SIP configuration, specified session agents, and specified SIP interfaces. This forces users who have a large group of remote agents to create a large number of session agents and SIP interfaces.

With this feature implemented, however, you can group remote agents into one or more realms on which to apply session constraints.

To enable sessions constraints on a per realm basis:

1. **constraint-name**—Enter the name of the constraint you want to use for this realm. You set up in the session-constraints confiuration.

UDP Checksum Generation Configuration

To enable UDP checksum generation for a realm:

1. **generate-udp-checksum**—Enable this parameter to generate a UDP checksum for this outbound realm. The default is **disabled**. Valid values are:
   - enabled | disabled

Admission Control Configuration

You can set admission control based on bandwidth for each realm by setting the **max-bandwidth** parameter for the realm configuration. Details about admission control are covered in this guide's *Admission Control and QoS* chapter.

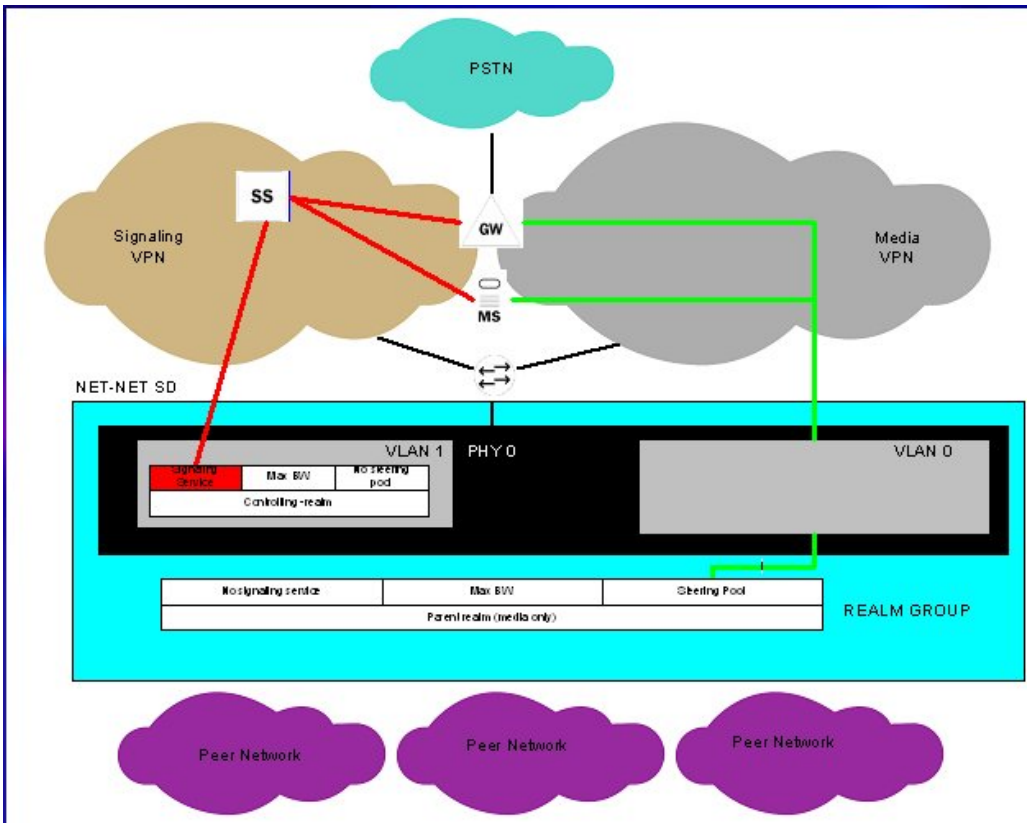
Reserved Parameters

In the ACLI, you do not need to configure the following parameters: **max-latency**, **max-jitter**, **max-packet-loss**, and **observ-window-size**.

Nested Realms

Configuring nested realms allows you to create backbone VPN separation for signaling and media. This means that you can put signaling and media on separate network interfaces, that the signaling and media VPN can have different address spaces, and that the parent realm has one media-only sub-realm.

The following figure shows the network architecture.



In addition, you can achieve enhanced scalability by using a shared service interface. A single service address is shared across many customers/peers, customer specific policies for bandwidth use and access control are preserved, and you can achieve fine-grained policy control.

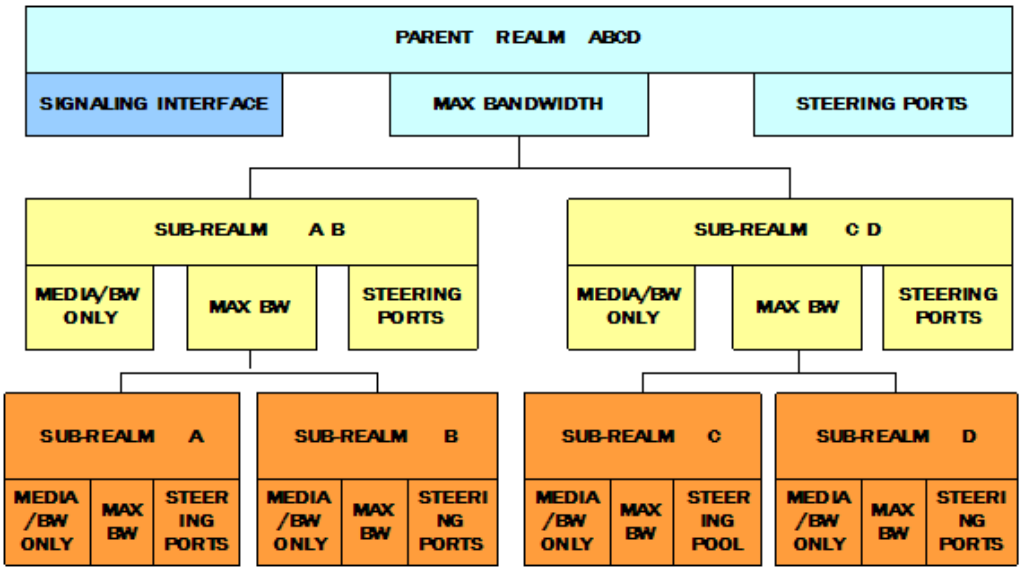
These benefits are achieved when you configure these types of realms:

- Realm group—A hierarchical nesting of realms identified by the name of the highest order realm.
- Controlling realm—A realms for which a signaling interface is configured. For example, you might configure these signaling interfaces in the following configurations: SIP-NAT or SIP port. Typically, this is the highest order realm for the parent realm in a realm group.
- Parent realm—A realm that has one or more child realms. A parent realm might also be the child realm of another realm group.
- Child realm—A realm that is associated with a single higher order parent realm. A child might also be the parent realm of another realm group. Child realms inherit all signaling and steering ports from higher order realms.
- Media-only realm—A realm for which there is no configured signaling interface directly associated. Media-only realms are nested within higher order realms.

As these definitions suggest, parent and child realms can be constructed so that there are multiple nesting levels. Lower order realms inherit the traits of the realms above them, including: signaling service interfaces, session translation tables, and steering pools.

Since realms inherit the traits of the realms above them in the hierarchy, you will probably want to map what realms should be parents and children before you start configuring them. These relationships are constructed through one parameter in the realm configuration that identifies the parent realm for the configuration. If you specify a parent realm, then the realm you are configuring becomes a child realm subject to the configured parameters you have established

for that parent. And since parent realms can themselves be children of other realm, it is important that you construct these relationships with care.



Configuring Nested Realms

When you are configuring nested realms, you can separate signaling and media by setting realm parameters in the SIP interface configuration and the steering ports configuration.

- The realm identifier you set in the SIP interface configuration labels the associated realm for signaling.
- The realm identifier you set in the steering ports configuration labels the associated realm for media.

Constructing a hierarchy of nested realms requires that you note which realms you want to handle signaling, and which you want to handle media.

In the SIP port configuration for the SIP interface, you will find an allow anonymous parameter that allows you to set certain access control measures. The table below outlines what each parameter means.

Allow Anonymous Parameter	Description
all	All anonymous connections allowed.
agents-only	Connections only allowed from configured session agents.
realm-prefix	Connections only allowed from addresses with the realm's address prefix and configured session agents.
registered	Connections allowed only from session agents and registered endpoints. (For SIP only, a REGISTER is allowed for any endpoint.)
register-prefix	Connections allowed only from session agent and registered endpoints. (For SIP only, a REGISTER is allowed for session agents and a matching realm prefix.)

Parent and Child Realm Configuration

To configure nested realms, you need to set parameters in the realm configuration.

To configure parent and child realms:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **parent-realm**—Enter the identifier of the realm you want to name as the parent. Configuring this parameter makes the realm you are currently configuring as the child of the parent you name. As such, the child realm is subject to the configured parameters for the parent.

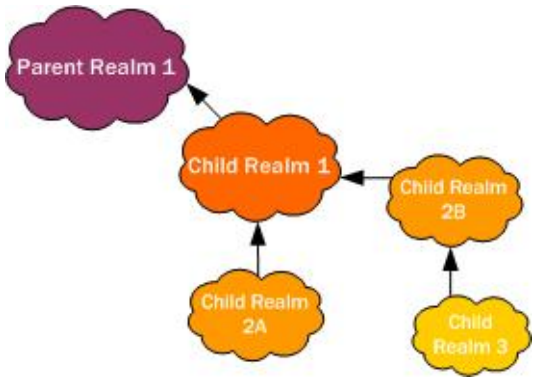
Aggregate Session Constraints Nested Realms

In addition to setting session constraints per realm for SIP sessions, you can also enable the Oracle USM to apply session constraints across nested realms. When you set up session constraints for a realm, those constraints apply only to the realm for which they are configured without consideration for its relationship either as parent or child to any other realms.

You can also, however, enable the Oracle USM to take nested realms into consideration when applying constraints. For example, if a call enters on a realm that has no constraints but its parent does, then the constraints for the parent are applied. This parameter is global and so applies to all realms on the system. For the specific realm the call uses and for all of its parents, the Oracle USM increments the counters upon successful completion of an inbound or outbound call.

In the following example, you can see one parent realm and its multiple nested, child realms. Now consider applying these realm constraints:

- Parent Realm 1—55 active sessions
- Child Realm 1—45 active sessions
- Child Realm 2A—30 active sessions
- Child Realm 2B—90 active sessions
- Child Realm 3—20 active sessions



Given the realm constraints outlined above, consider these examples of how global session constraints for realms. For example, a call enters the Oracle USM on Child Realm 2B, which has an unmet 90-session constraint set. Therefore, the Oracle USM allows the call based on Child Realm 2B. But the call also has to be within the constraints set for Child Realm 1 and Parent Realm 1. If the call fails to fall within the constraints for either of these two realms, then the Oracle USM rejects the call.

Impact to Other Session Constraints and Emergency Calls

You can set up session constraints in different places in your Oracle USM configuration. Since session agents and SIP interfaces also take session constraints, it is import to remember the order in which the Oracle USM applies them:

1. Session agent session constraints
2. Realm session constraints (including parent realms)
3. SIP interface session constraints

   Emergency and priority calls for each of these is exempt from session constraints. That is, any call coming into the Oracle USM marked priority is processed.

Session Contraints Configuration

You enabled use of session constraints for nested realms across the entire system by setting the **nested-realms-stats** parameter in the session router configuration to **enabled**.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-router** and press Enter.

   ```
   ORACLE(session-router)# session-router
   ORACLE(session-router-config)#
   ```

4. **nested-realms-stats**—Change this parameter from **disabled** (default) to **enabled** if you want the Oracle USM to apply session constraints across all nested realms (realms that are children to other realms)
5. Save and activate your configuration.

Realm-Based Packet Marking

The Oracle USM supports TOS/DiffServ functions that allow you to

- Set up realm-based packet marking by media type, either audio-voice or video
- Set up realm-based packet marking for signaling

Upstream devices use these markings to classify traffic in order to determine the priority level of treatment it will receive.

About TOS DiffServ

TOS and DiffServ are two different mechanisms used to achieve QoS in enterprise and service provider networks; they are two different ways of marking traffic to indicate its priority to upstream devices in the network.

For more information about TOS (packet) marking, refer to:

- IETF RFC 1349 (http://www.ietf.org/rfc/rfc1349.txt)

For more information about DiffServ, refer to:

- IETF RFC 2474 (http://www.ietf.org/rfc/rfc2474.txt)
- IETF RFC 2475 (http://www.ietf.org/rfc/rfc2475.txt).

ToS Byte

The TOS byte format is as follows:

The TOS byte is broken down into three components:

- Precedence—The most used component of the TOS byte, the precedence component is defined by three bits. There are eight possible precedence values ranging from 000 (decimal 0) through 111 (decimal 7). Generally, a precedence value of 000 refers to the lowest priority traffic, and a precedence value of 111 refers to the highest priority traffic.
- TOS—The TOS component is defined by four bits, although these bits are rarely used.
- MBZ—The must be zero (MBZ) component of the TOS byte is never used.

DiffServ Byte

Given that the TOS byte was rarely used, the IETF redefined it and in doing so created the DiffServ byte.

The DiffServ byte format is as follows:

The DiffServ codepoint value is six bits long, compared to the three-bit-long TOS byte's precedence component. Given the increased bit length, DiffServ codepoints can range from 000000 (decimal 0) to 111111 (decimal 63).

> **Note:**
>
> By default, DiffServ codepoint mappings map exactly to the precedence component priorities of the original TOS byte specification.

Packet Marking for Media

You can set the TOS/DiffServ values that define an individual type or class of service for a given realm. In addition, you can specify:

- One or more audio media types for SIP
- One or more video media types for SIP
- Both audio and video media types for SIP

For all incoming SIP requests, the media type is determined by negotiation or by preferred codec. SIP media types are determined by the SDP.

Configuring Packet Marking by Media Type

This section describes how to set up the media policy configuration that you need for this feature, and then how to apply it to a realm.

These are the ACLI parameters that you set for the media policy:

```
name         media policy name
tos-settings  list of TOS settings
```

> **Note:**
>
> The **media-policy > tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

This is the ACLI parameter that you set for the realm:

```
media-policy default media policy name
```

Packet Marking Configuration

To set up a media policy configuration to mark audio-voice or video packets:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# media-policy
   ORACLE(media-policy)#
   ```

   From this point, you can configure media policy parameters. To view all configuration parameters for media profiles, enter a **?** at the system prompt.

4. Type **media-policy** and press Enter.

   ```
   ORACLE(media-manager)# media-policy
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

5. **name**—Create a reference name for this policy and press Enter.

6. Type **tos-settings** and press Enter.

   ```
   ORACLE(media-policy)# tos-settings
   ```

7. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-type message
   ```

8. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-sub-type sip
   ```

9. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

   ```
   ORACLE(tos-settings)# media-attributes sendonly sendrecv
   ```

10. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:

    - 0x00 to 0xFF.

      ```
      ORACLE(tos-settings)# tos-value 0xF0
      ```

11. Save and activate your configuration.

Applying a Media Policy to a Realm

To apply a media policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm
ORACLE(realm)#
```

4. **media-policy**—Enter the unique name of the media policy you want to apply to this realm.

Signaling Packet Marking Configuration

ToS marking for signaling requires you to configure a media policy and set the name of the media policy in the appropriate realm configuration.

This section shows you how to configure packet marking for signaling.

Configuring a Media Policy for Signaling Packet Marking

To set up a media policy configuration to mark signaling packets:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# media-manager
```

3. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# media-policy
ORACLE(media-policy)#
```

From this point, you can configure media policy parameters. To view all media policy configuration parameters, enter a **?** at the system prompt.

4. Type **media-policy** and press Enter.

```
ORACLE(media-manager)# media-policy
```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

5. **name**—Create a reference name for this policy and press Enter.
6. Type **tos-settings** and press Enter.

```
ORACLE(media-policy)# tos-settings
```

> **Note:**
>
> The **media-policy > tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

7. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image, message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-type message
```

8. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-sub-type sip
   ```

9. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

   ```
   ORACLE(tos-settings)# media-attributes sendonly sendrecv
   ```

10. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:

    - 0x00 to 0xFF.

      ```
      ORACLE(tos-settings)# tos-value 0xF0
      ```

11. Save and activate your configuration.

Applying a Media Policy to a Realm

To apply a media policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm
   ORACLE(realm)#
   ```

4. **media-policy**—Enter the unique name of the media policy you want to apply to this realm.

Using Class Profile for Packet Marking

Class profile provides an additional means of ToS marking, but only for limited circumstances. Use class-profile only if you are marking ToS on traffic destined for a specific To address, and when media-policy is not used on the same realm. Using media-policy for ToS marking is, by far, more common.

To configure a class profile, you prepare your desired media policy, create the class profile referencing the media policy and the To address, and set the name of the class profile in the appropriate realm configuration.

Class Profile and Class Policy Configuration

This section shows you how to configure packet marking using a class profile.

To configure the class profile and class policy:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **class-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# class-profile
ORACLE(class-profile)#
```

4. Type **policy** and press Enter to begin configuring the class policy.

```
ORACLE(class-profile)# policy
```

From this point, you can configure class policy parameters. To view all class policy configuration parameters, enter a **?** at the system prompt.

5. **profile-name**—Enter the unique name of the class policy. When you apply a class profile to a realm configuration, you use this value.
6. **to-address**—Enter a list of addresses to match to incoming traffic for marking. You can use E.164 addresses, a host domain address, or use an asterisk (*) to set all host domain addresses.
7. **media-policy**—Enter the name of the media policy you want to apply to this class policy.

Applying a Class Policy to a Realm

To apply a class policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# media-manager
```

3. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm
ORACLE(realm)#
```

4. **class-profile**—Enter the name if the class profile to apply to this realm. This is the name you set in the **profile-name** parameter of the class-policy configuration.

Differentiated Services for DNS and ENUM

The Oracle USM can mark DNS/ENUM packets with a configurable differentiated services code point (DSCP).

Certain service providers mandate support for Differentiated Services (DS) on all traffic streams exiting any network element. This mandate requires that network elements function as a DS marker — that is as a device that sets the Distributed Services Codepoint (DSCP) in the Differentiated Services field of the IP header.

Previous software releases provided the capabilities to mark standard SIP and Real-Time Protocol (RTP) messages. Release S-CX6.4.0M3 adds the capability to mark DNS and ENUM queries.

For basic information about DS, refer to:

- The *ACLI Configuration Guide* Realm-Based Packet Marking topic in the Realms and Nested Realms chapter, which provides information on currently supported DS marking of SIP and RTP packets
- *IETF RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers* (http://www.ietf.org/rfc/rfc2474.txt)
- *IETF RFC 2475, An Architecture for Differentiated Services* (http://www.ietf.org/rfc/rfc2475.txt).

DS provides a mechanism to define and deliver multiple and unique service classifications that can be offered by a service provider. Specific service classifications are identified by a DSCP, essentially a numeric index. The DSCP maps to a per-hop-behavior (PHB) that defines an associated service class. PHBs are generally defined in terms of call admission controls, packet drop criteria, and queue admission algorithms. In theory, DS supports 64 distinct classifications. In practice, however, network offerings generally consist of a much smaller suite, which typically includes:

- Default PHB - required best effort service — defined in RFC 2474
- Expedited Forwarding (EF) PHB - low-loss, low-latency — defined in RFC 3246, An Expedited Forwarding PHB
- Assured Forwarding (AF) PHB - assured delivery within subscriber limits — defined in *RFC 2597, Assured Forwarding PHB Group*, and *RFC 3260, New Terminology and Clarifications for Diffserv*
- Class Selector PHBs - maintain compatibility with previous TOS (type of service) precedence usage — defined in RFC 2474

Marking of DNS and ENUM queries requires the creation of a Differentiated Services media policy, and the assignment of such a policy to a specific realm.

Differentiated Services for DNS and ENUM Configuration

To create a Differentiated Services media policy:

1. From Superuser mode, use the following ACLI command sequence to move to media-policy configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-policy
ORACLE(media-policy)#
```

2. Use the required **name** parameter to provide a unique identifier for this media-policy instance.

```
ORACLE(media-policy)# name diffServeDnsEnum
ORACLE(media-policy)#
```

3. Use the tos-settings command to move to tos-settings configuration mode.

```
ORACLE(media-policy)# tos-settings
ORACLE(tos-settings)#
```

> **Note:**
>
> The **media-policy > tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

4. Use the required media-type parameter to identify the packet-type subject to Diffentiated Services marking.

   For DNS/ENUM packets, use message/dns.

   ```
   ORACLE(tos-settings)# media-type message/dns
   ORACLE(tos-settings)#
   ```

5. Use the required tos-value parameter to specify the 6-bit DSCP to be included in the Differentiated Services field of the IP header.

   Specify the DSCP as an integer (within the range from 0 to 63). The DSCP can be expressed in either decimal or hexadecimal format.

   ```
   ORACLE(tos-settings)# tos-value 0x30
   ORACLE(tos-settings)#
   ```

6. Other displayed parameters, media-attributes and media-sub-type, can be safely ignored.

7. Use the done and exit commands to complete Differentiated Services media policy configuration and return to media-policy configuration-mode.

   ```
   ORACLE(tos-settings)# done
   tos-settings
           media-type                  message/dns
           media-sub-type
           tos-value                   0x30
           media-attributes
   ORACLE(tos-settings)# exit
   ORACLE(media-policy)#
   ```

8. If necessary, repeat steps 2 through 7 to create additional Differentiated Services media policies.

9. Assign this media policy to the target realm via the **media-policy** parameter in a **realm-config** object.

SIP-SDP DCSP Marking ToS Bit Manipulation

Used to indicate priority and type of requested service to devices in the network, type of service (TOS) information is included as a set of four-bit flags in the IP header. Each bit has a different purpose, and only one bit at a time can be set: There can be no combinations. Available network services are:

- Minimum delay—Used when latency is most important
- Maximum throughput—Used when the volume of transmitted data in any period of time is important
- Maximum reliability—Used when it is important to assure that data arrives at its destination without requiring retransmission
- Minimum cost—Used when it is most important to minimize data transmission costs

The Oracle USM's support for type of service (TOS allows you to base classification on the media type as well as the media subtype. In prior releases, you can configure the Oracle USM to mark TOS bits on outgoing packets using a media policy. Supported media types include audio, video, application, data, image, text, and message; supported protocol types are H.225, H.245, and SIP. Note that, although H.225 and H.245 are not part of any IANA types, they are special cases (special subtypes) of message for the Oracle USM. When these criteria are met for an outgoing packet, the Oracle USM applies the TOS settings to the IP header. The

augmented application of TOS takes matching on media type or protocol and expands it to match on media type, media-sub-type, and media attributes.

The new flexibility of this feature resolves issues when, for example, a customer needs to differentiate between TV-phone and video streaming. While both TV-phone and video streaming have the attribute “media=video,” TV-phone streaming has “direction=sendrcv” prioritized at a high level and video has direction=sendonly or recvonly with middle level priority. The Oracle USM can provide the appropriate marking required to differentiate the types of traffic.

In the media policy, the **tos-values** parameter accepts values that allow you to create any media type combination allowed by IANA standards. This is a dynamic process because theOracle USM generates matching criteria directly from messages.

The new configuration takes a media type value of any of these: audio, example, image, message, model, multipart, text, and video. It also takes a media sub-type of any value specified for the media type by IANA; however, support for T.38 must be entered exactly as **t.38** (rather than t38). Using these values, the Oracle USM creates a value Based on a combination of these values, the Oracle USM applies TOS settings.

You also configure the TOS value to be applied, and the media attributes you want to match.

You can have multiple groups of TOS settings for a media policy.

ToS Bit Manipulation Configuration

This section provides instructions for how to configure TOS bit manipulation on your Oracle USM.

To configure TOS bit manipulation:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-policy** and press Enter.

   ```
   ORACLE(media-manager)# media-policy
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **name**—Create a reference name for this policy and press Enter.
5. Type **tos-settings** and press Enter.

   ```
   ORACLE(media-policy)# tos-settings
   ```

   > **Note:**
   >
   > The **media-policy > tos-settings** parameter is not RTC supported and a reboot is required for these updates to take affect.

6. **media-type**—Enter the media type that you want to use for this group of TOS settings. You can enter any of the IANA-defined media types for this value: audio, example, image,

message, model, multipart, text, and video. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

```
ORACLE(tos-settings)# media-type message
```

7. **media-sub-type**—Enter the media sub-type you want to use for the media type. This value can be any of the sub-types that IANA defines for a specific media type. This value is not case-sensitive and can be up to 255 characters in length; it has no default.

   ```
   ORACLE(tos-settings)# media-sub-type sip
   ```

8. **media-attributes**—Enter the media attribute that will match in the SDP. This parameter is a list, so you can enter more than one value. The values are case-sensitive and can be up to 255 characters in length. This parameter has no default.

   If you enter more than one media attribute value in the list, then you must enclose your entry in quotation marks ().

   ```
   ORACLE(tos-settings)# media-attributes sendonly sendrecv
   ```

9. **tos-value**—Enter the TOS value you want applied for matching traffic. This value is a decimal or hexidecimal value. The valid range is:

   - 0x00 to 0xFF.

     ```
     ORACLE(tos-settings)# tos-value 0xF0
     ```

10. Save and activate your configuration.

DSCP Marking for MSRP and Media Over TCP

The Oracle USM supports Differentiated Services Code Point (DSCP) marking of MSRP and Media over TCP traffic. This feature may be used for MSRP traffic in both B2BUA and non-B2BUA modes.

In order to configure the Oracle USM to mark MSRP or media over TCP packets with a value in the IP header's DSCP field, create a **media-manager** > **media-policy** > **tos-settings** configuration element and set the **media-type** parameter to **message**. This has the effect of marking all traffic described by prior SDP with m=message with this DSCP value.

Note that setting **media-type** to **message** can potentially mark a large range of traffic. For other common traffic to remain marked differently than marked MSRP, you need to create additional **tos-settings** configuration elements with valid **media-sub-type** configurations. The following values may be provisioned in the **media-sub-type** parameter to mark these other types of traffic differently (with an accompanying **tos-value**).

- sip
- li
- dns

SDP Alternate Connectivity

The Oracle USM can create an egress-side SDP offer containing both IPv4 and IPv6 media addresses via a mechanism which allows multiple IP addresses, of different address families (i.e., IPv4 & IPv6) in the same SDP offer. Our implementation is based on the RFC draft "draft-boucadair-mmusic-altc-09".

Each realm on the Oracle USM can be configured with an alternate family realm on which to receive media in the alt family realm parameter in the realm config. As deployed, one realm

will be IPv4, and the alternate will be IPv6. The Oracle USM creates the outbound INVITE with IPv4 and IPv6 addresses to accept the media, each in an a=altc: line and each in its own realm. The IP addresses inserted into the a=altc: line are from the egress realm's and alt-realm-family realm's steering pools. Observe in the image how the red lines indicate the complementary, alternate realms.

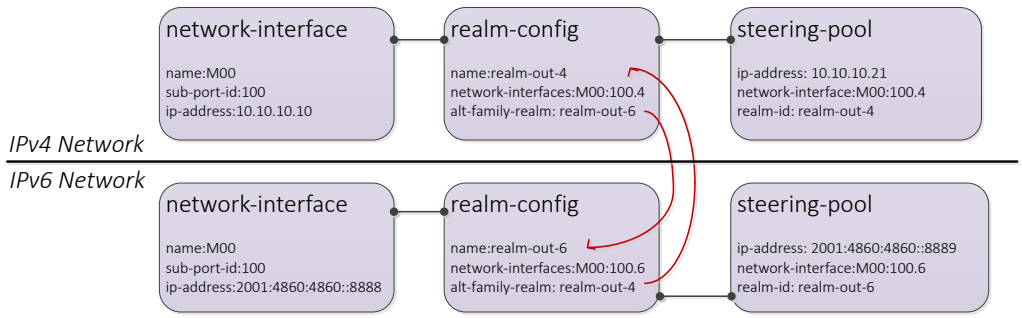
You can configure the order in which the a=altc: lines appear in the SDP in the pref-address-type parameter in the realm-config. This parameter can be set to

- IPv4 - SDP contains the IPv4 address first
- IPv6 - SDP contains the IPv6 address first
- NONE - SDP contains the native address family of the egress realm first

In the 200OK to the INVITE, the callee chooses either the IPv6 or IPv4 address to use for the call's media transport between itself and Oracle USM. After the Oracle USM receives the 200OK, the chosen flow is installed, and the unused socket is discarded.

For two realms from different address families to share the same physical interface and vlan, you use a .4 or .6 tag in the network-interface reference. When IPv4 and IPv6 realms share the same network-interface and VLAN, you identify them by realm name and network-interface configured as:

- IPv4 - <phy-interface>:<vlan>.4
- IPv6 - <phy-interface>:<vlan>.6



If the INVITE's egress realm is IPv6, pref-address-type = NONE, the outbound SDP has these a=altc: lines:

```
a=altc:1 IPv6 2001:4860:4860::8889 20001
a=altc:2 IPv4 10.10.10.21 20001
```

If the INVITE's egress realm is IPv6, pref-address-type = IPv4, the outbound SDP has these a=altc: lines:

```
a=altc:1 IPv4 10.10.10.21 20001
a=altc:2 IPv6 2001:4860:4860::8889 20001
```

SDP Alternate connectivity supports B2B and hairpin call scenarios. SDP Alternate connectivity also supports singleterm, B2B, and hairpin call scenarios.

When providing SDP alternate connectivity for SRTP traffic, in the security policy configuration element, the network-interface parameter's value must be configured with a .4 or .6 suffix to indicate IPv4 or IPv6 network, respectively.

SDP Alternate Connectivity Configuration

To configure SDP alternate connectivity:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **alt-realm-family** — Enter the realm name of the alternate realm, from which to use an IP address in the other address family. If this parameter is within an IPv4 realm configuration, you will enter an IPv6 realm name.
4. **pref-address-type** — Set the order in which the a=altc: lines suggest preference. Valid values are:
   - none — address family type of egress realm signaling
   - ipv4 — IPv4 realm/address first
   - ipv6 — IPv6 realm/address first
5. Type **done** to save your configuration.

Steering Pools

Steering pools define sets of ports that are used for steering media flows through the Oracle USM. These selected ports are used to modify the SDP to cause receiving session agents to direct their media toward this system. Media can be sent along the best quality path using these addresses and ports instead of traversing the shortest path or the BGP-4 path.

For example, when the Oracle USM is communicating with a SIP device in a specific realm defined by a steering pool, it uses the IP address and port number from the steering pool's range of ports to direct the media. The port the Oracle USM chooses to use is identified in the SDP part of the message.

> **Note:**
>
> The values entered in the steering pool are used when the system provides NAT, PAT, and VLAN translation.

Configuration Overview

To plan steering pool ranges, take into account the total sessions available on the box, determine how many ports these sessions will use per media stream, and assign that number of

ports to all of the steering pools on your Oracle USM. For example, if your Oracle USM can accommodate 500 sessions and each session typically uses 2 ports, you would assign 1000 ports to each steering pool. This strategy provides for a maximum number of ports for potential use, without using extra resources on ports your Oracle USM will never use.

The following table lists the steering pool parameters you need to configure:

Parameter	Description
IP address	IPv4 address of the steering pool.
start port	Port number that begins the range of ports available to the steering pool. You must define this port to enable the system to perform media steering and NAT operations.
end port	Port number that ends the range of ports available to the steering pool. You must define this port to enable the system to perform media steering and NAT operations.
realm id	Identifies the steering pool's realm. The steering pool is restricted to only the flows that originate from this realm.

Note:

The combination of entries for IP address, start port, and realm ID must be unique in each steering pool. You cannot use the same values for multiple steering pools.

Each bidirectional media stream in a session uses two steering ports, one in each realm (with the exception of audio/video calls that consume four ports). You can configure the start and end port values to provide admission control. If all of the ports in all of the steering pools defined for a given realm are in use, no additional flows/sessions can be established to/from the realm of the steering pool.

Steering Pool Configuration

To configure a steering pool:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **steering-pool** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# steering-pool
   ORACLE(steering-pool)#
   ```

4. **ip-address**—Enter the target IPv4 address of the steering pool in IP address format. For example:

   ```
   192.168.0.11
   ```

5. **start-port**—Enter the start port value that begins the range of ports available to this steering pool. The default is **0**. The valid range is:

- Minimum—0
- Maximum—65535

  You must enter a valid port number or the steering pool will not function properly.

6. **end-port**—Enter the end port value that ends the range of ports available to this steering pool. The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

     You must enter a valid port number or the steering pool will not function properly.

7. **realm-id**—Enter the realm ID to identify the steering pool's realm, following the name format. The value you enter here must correspond to the value you entered as the identifier (name of the realm) when you configured the realm. For example:

   ```
   peer-1
   ```

   This steering pool is restricted to flows that originate from this realm.

   The following example shows the configuration of a steering pool that

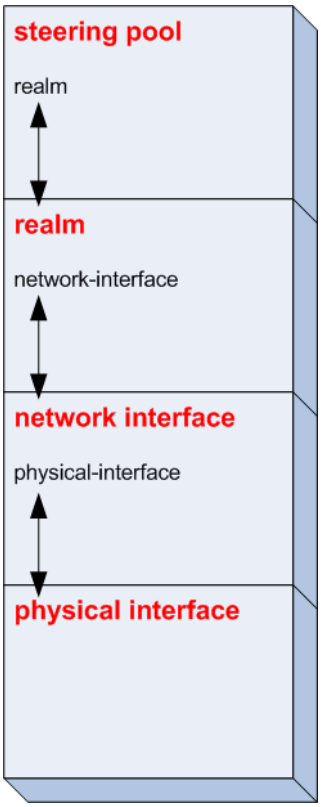
   ```
   steering-pool
           ip-address                    192.168.0.11
           start-port                    20000
           end-port                      21000
           realm-id                      peer-1
           last-modified-date            2005-03-04 00:35:22
   ```

Multiple Interface Realms

The multi-interface realm feature lets you group multiple network interfaces to aggregate their bandwidth for media flows. In effect, this feature lets you use the total throughput of the available physical interfaces on your Oracle USM for a single realm. Multi-interface realms are implemented by creating multiple steering pools, each on an individual network interface, that all reference a single realm.

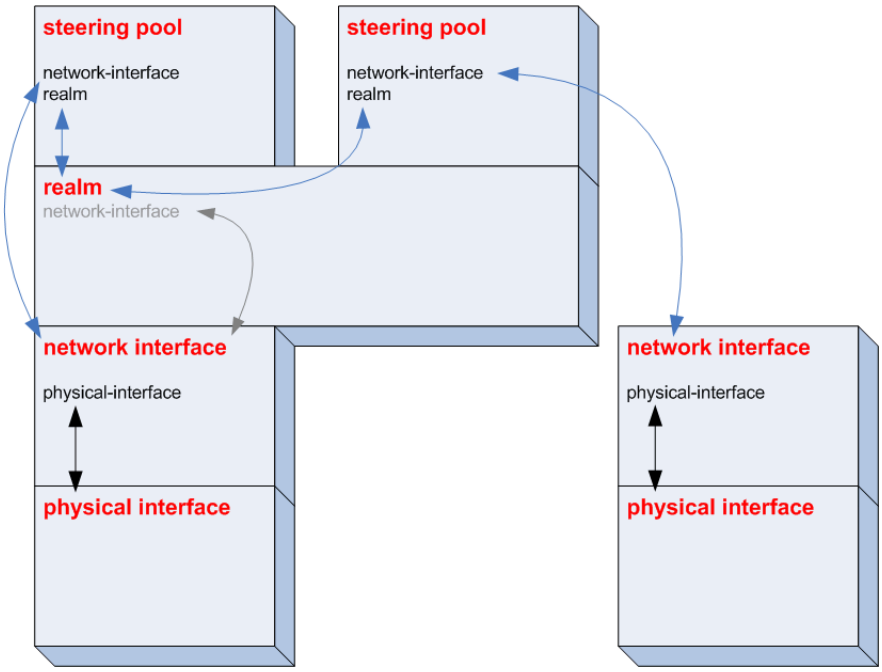
Of course, you can not to use this feature and configure your Oracle USM to create a standard one-realm to one-network interface configuration.

Without using multiple interface realms, the basic hierarchical configuration of the Oracle USM from the physical interface through the media steering pool looks like this:

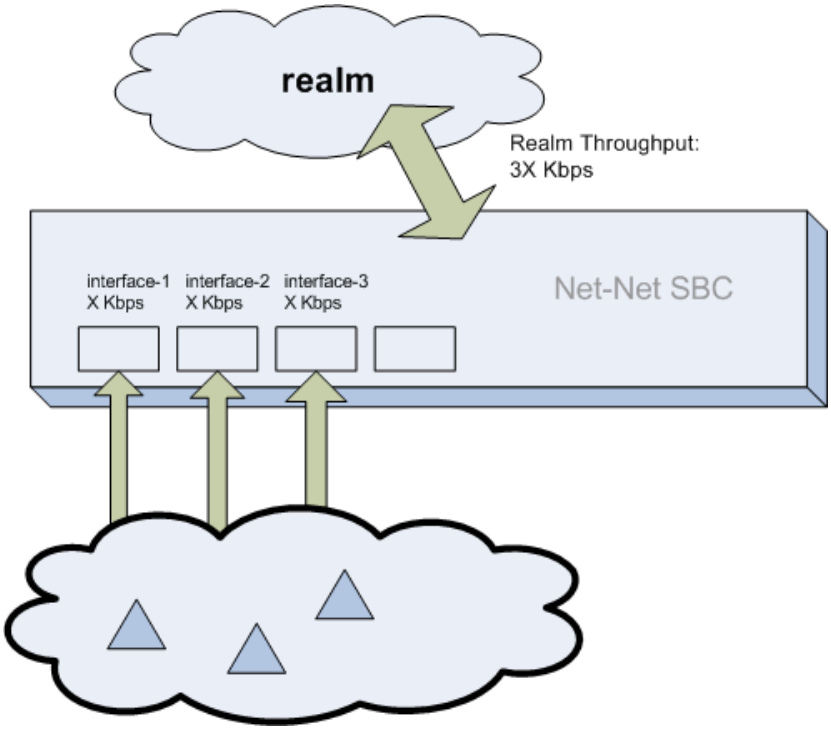


In this model, one (non-channelized) network interface exists on a physical interface. One realm exists on one network interface. One or more steering pools can exist on one realm. Within each higher level configuration element exists a parameter that references a lower level configuration element in the Oracle USM's logical network model.

The multi-interface realm feature directs media traffic entering and exiting multiple network interfaces in and out of a single realm. Since all the steering pools belong to the same realm, their assigned network interfaces all feed into the same realm as well. The following diagram shows the relationship in the new logical model:



The advantage of using multi-interface realms is the ability to aggregate the bandwidth available to multiple network interfaces for a larger-than-previously-available total bandwidth for a realm. In the illustration below, three physical interfaces each have X Kbps of bandwidth. The total bandwidth available to the realm with multiple network interfaces is now 3X the bandwidth. (In practical usage, interface-1 only contributes X - VoIP Signaling to the total media bandwidth available into the realm.)



Steering Pool Port Allocation

Every steering pool you create includes its own range of ports for media flows. The total number of ports in all the steering pools that feed into one realm are available for calls in and out of the realm.

Steering pool ports for a given realm are assigned to media flows sequentially. When the first call enters the Oracle USM after start-up, it is assigned the first ports on the first steering pool that you configured. New calls are assigned to ports sequentially in the first steering pool. When all ports from the first steering pool are exhausted, the Oracle USM uses ports from the next configured steering pool. This continues until the last port on the last configured steering pool is used.

After the final port is used for the first time, the next port chosen is the one first returned as empty from the full list of ports in all the steering pools. As media flows are terminated, the ports they used are returned to the realm's full steering pool. In this way, after initially exhausting all ports, the realm takes new, returned, ports from the pool in a least last used manner.

When a call enters the Oracle USM, the signaling application allocates a port from all of the eligible steering pools that will be used for the call. Once a port is chosen, the Oracle USM checks if the steering pool that the port is from has a defined network interface. If it does, the call is set up on the corresponding network interface. If a network interface is not defined for that steering pool, the network interface defined for the realm is used.

Network Interface Configuration

This section explains how to configure your Oracle USM to use multiple interface realms.

You must first configure multiple physical interfaces and multiple network interfaces on your Oracle USM.

To configure the realm configuration for multi-interface realms.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   From this point, you can configure a realm that will span multiple network interfaces.

4. **network-interfaces**—Enter the name of the network interface where the signaling traffic for this realm will be received.

Creating Steering Pools for Multiple Interface Realms

To configure steering pools for multi-interface realms:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-manager path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **steering-pool** and press Enter. The system prompt changes.

   ```
   ORACLE(media-manager)# steering-pool
   ORACLE(steering-pool)#
   ```

   From this point, you can configure steering pools which collectively bridge the multiple network interfaces they are connected to.

4. **ip-address**—Enter the IP address of the first steering pool on the first network interface.

   This IP address must correspond to an IP address within the subnet of a network interface you have already configured.

   This IP can not exist on a network interface other than the one you configure in the **network-interface** parameter.

5. **start-port**—Enter the beginning port number of the port range for this steering pool. The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

6. **end-port**—Enter the ending port number of the port range for this steering pool. The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

7. **realm-id**—Enter the name of the realm which this steering pool directs its media traffic toward.

8. **network-interface**—Enter the name of the network interface you want this steering pool to direct its media toward. This parameter will match a **name** parameter in the network-interface configuration element. If you do not configure this parameter, you can only assign a realm to a single network interface, as the behavior was in all Oracle USM Software releases pre- 2.1.

9. Create additional steering pools on this and on other network interfaces as needed. Remember to type **done** when you are finished configuring each new steering pool.

Media over TCP

The Oracle USM now supports RFC 4145 (TCP-Based Media Transport in the SDP), also called TCP Bearer support. Media over TCP can be used to support applications that use TCP for bearer path transport.

RFC 4145 adds two new attributes, setup and connection, to SDP messages. The setup attribute indicates which end of the TCP connection should initiate the connection. The connection attribute indicates whether an existing TCP connection should be used or if a new TCP connection should be setup during re-negotiation. RFC 4145 follows the offer/answer model specified in RFC3264. An example of the SDP offer message from the end point 192.0.2.2 as per RFC4145 is as given below:

```
m=image 54111 TCP t38
c=IN IP4 192.0.2.2
a=setup:passive
a=connection:new
```

This offer message indicates the availability of t38 fax session at port 54111 which runs over TCP. Oracle USM does not take an active part in the application-layer communication between each endpoint.

The Oracle USM provides the means to set up the end-to-end TCP flow by creating the TCP/IP path based on the information learned in the SDP offer/answer process.

TCP Bearer Conditions

The following conditions are applicable to the Oracle USM's support of RFC 4145.

1. The Oracle USM can not provide media-over-TCP for HNT scenarios (endpoints behind a NAT).
2. When media is released into the network, the TCP packets do not traverse the Oracle USM because nNo TCP bearer connection is created.
3. The Oracle USM does not inspect the setup and connection attributes in the SDP message since the TCP packets transparently pass through the Oracle USM. These SDP attributes are forwarded to the other endpoint. It is the other endpoint's responsibility to act accordingly.
4. After the Oracle USMreceives a SYN packet, it acts as a pure pass through for that TCP connection and ignores all further TCP handshake messages including FIN and RST. The flow will only be torn down in the following instances:
   - The TCP initial guard timer, TCP subsequent guard timer, or the TCP flow time limit timer expire for that flow.
   - The whole SIP session is torn down.

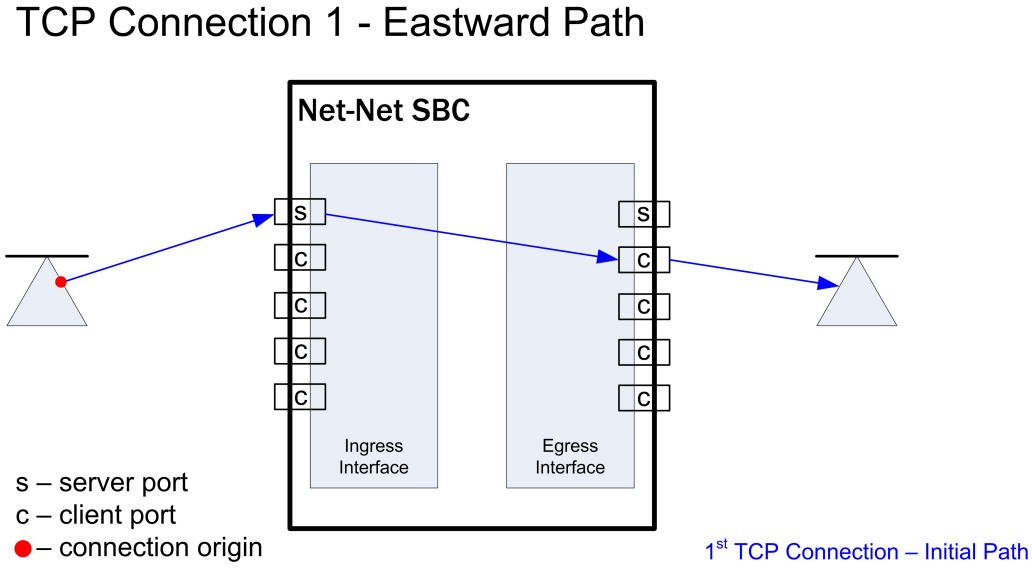
TCP Port Selection

When a call is first set up, the Oracle USM inspects the SDP message's m-line to see if any media will be transported via TCP. If the SDP message indicates that some content will use TCP, the Oracle USM allocates a configured number of steering ports for the media-over-TCP traffic. These TCP media ports are taken from the each realm's steering pool.

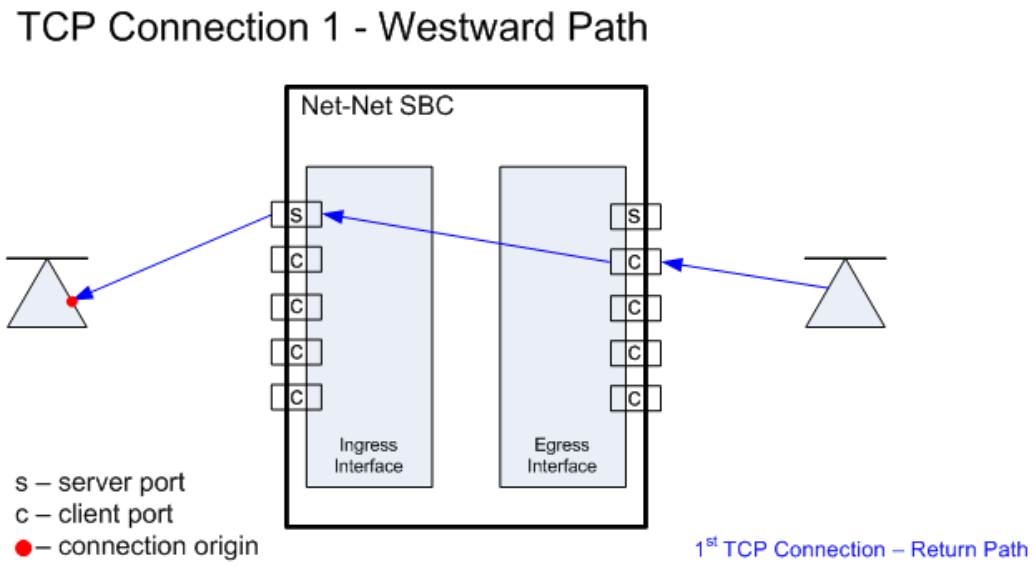
Each endpoint can initiate up to four end-to-end TCP flows between itself and the other endpoint. The Oracle USM assigns one port to receive the initial TCP packet (server port), and one to four ports assigned to send TCP traffic (client ports) to the receiving side of the TCP flow. The number of TCP flows for each call is configured globally.

In order to configure the Oracle USM to facilitate and support this process, you need to specify the number of ports per side of the call that can transport discrete TCP flows. You can configure one to four ports/flows. For configuration purposes, the Oracle USM counts this number as inclusive of the server port. Therefore if you want the Oracle USM to provide a maximum of one end-to-end TCP flow, you have to configure two TCP ports; one to receive, and one to send. The receiving port (server) is reused to set up every flow, but the sending port (client) is discrete per flow. For example: for 2 flows in each direction, set the configuration to 3 TCP ports per flow; for 3 flows in each direction, set the configuration to 4 TCP ports per flow, etc.

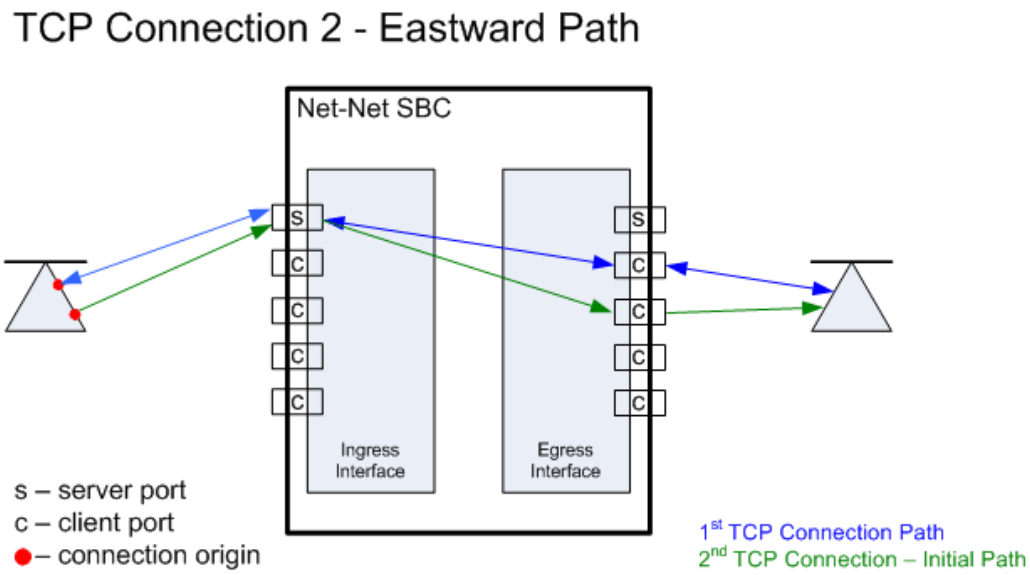
The server port is used for initiating a new TCP connection. An endpoint sends the first packet to a server port on the ingress interface. The packet is forwarded out of the Oracle USM through a client port on the egress interface toward an endpoint:



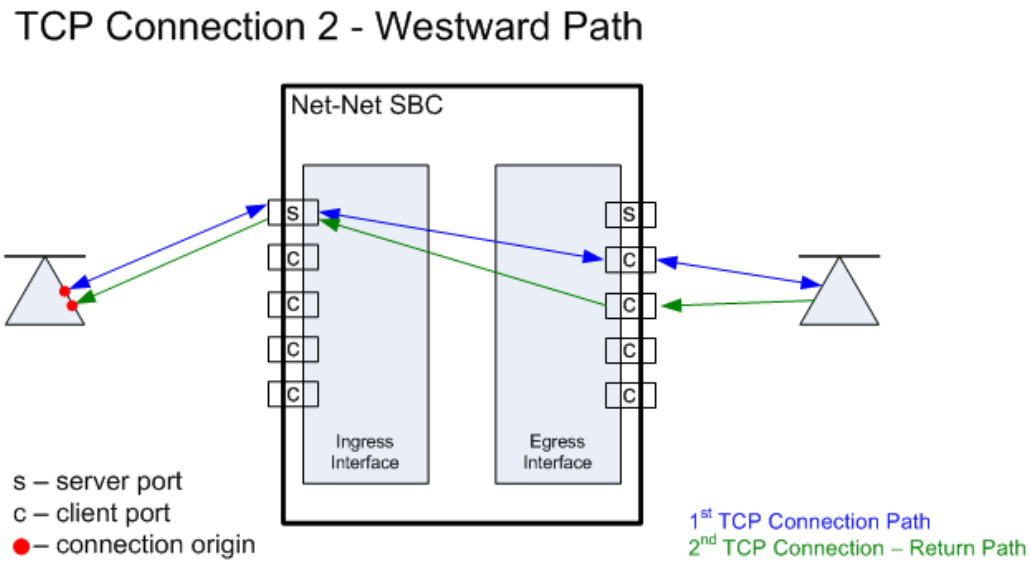
The endpoint responds back to the client port on the egress interface. This message traverses the Oracle USM and is forwarded out of the server port on the ingress interface where the initial packet was sent. The remainder of the TCP flow uses the server and client port pair as a tunnel through the Oracle USM:



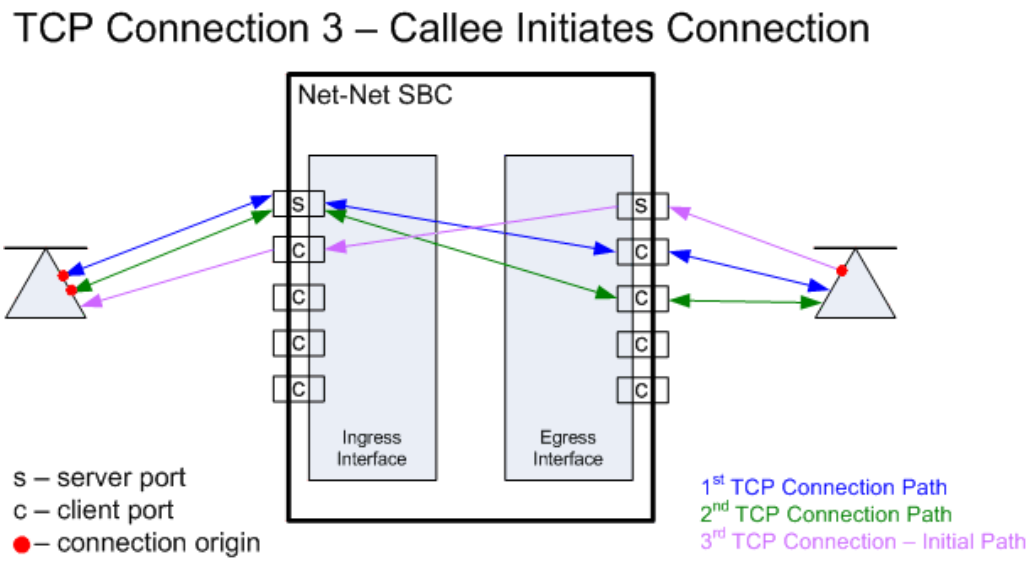
When the second TCP connection is set up in the same direction as in the first example, the first packet is still received on the server port of the ingress interface. The next unused client port is chosen for the packet to exit the Oracle USM:



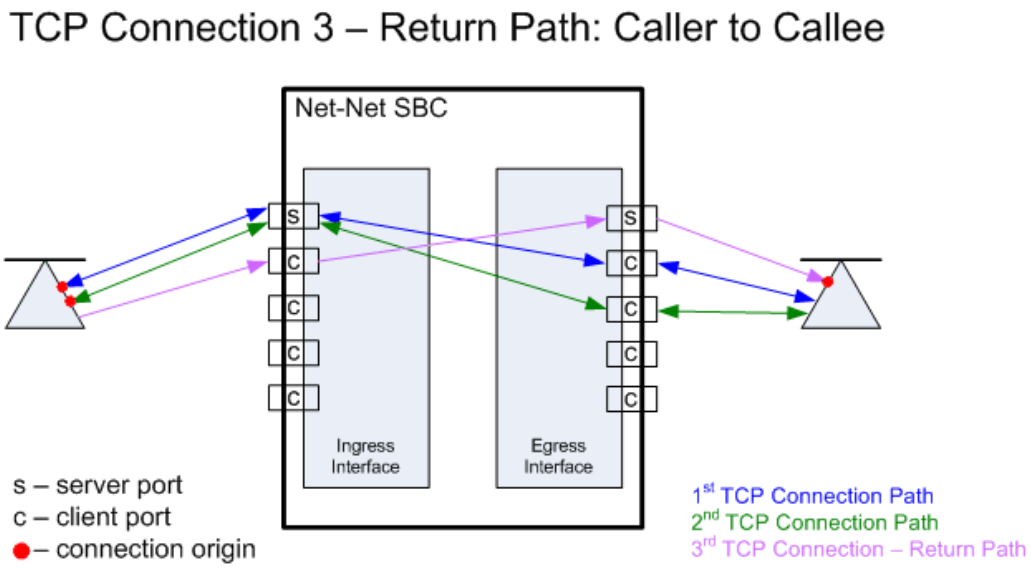
The response takes the same path back to the caller. The remainder of the second TCP connection uses this established path:



When the callee initiates a TCP connection, it must send its initial traffic to the server port on its Oracle USM ingress interface. The packet is forwarded out of the first free client port on the egress side of this TCP connection toward the caller.



The caller's response takes the same path back to the callee that initiated this TCP connection. The remainder of the third TCP connection uses this established path.

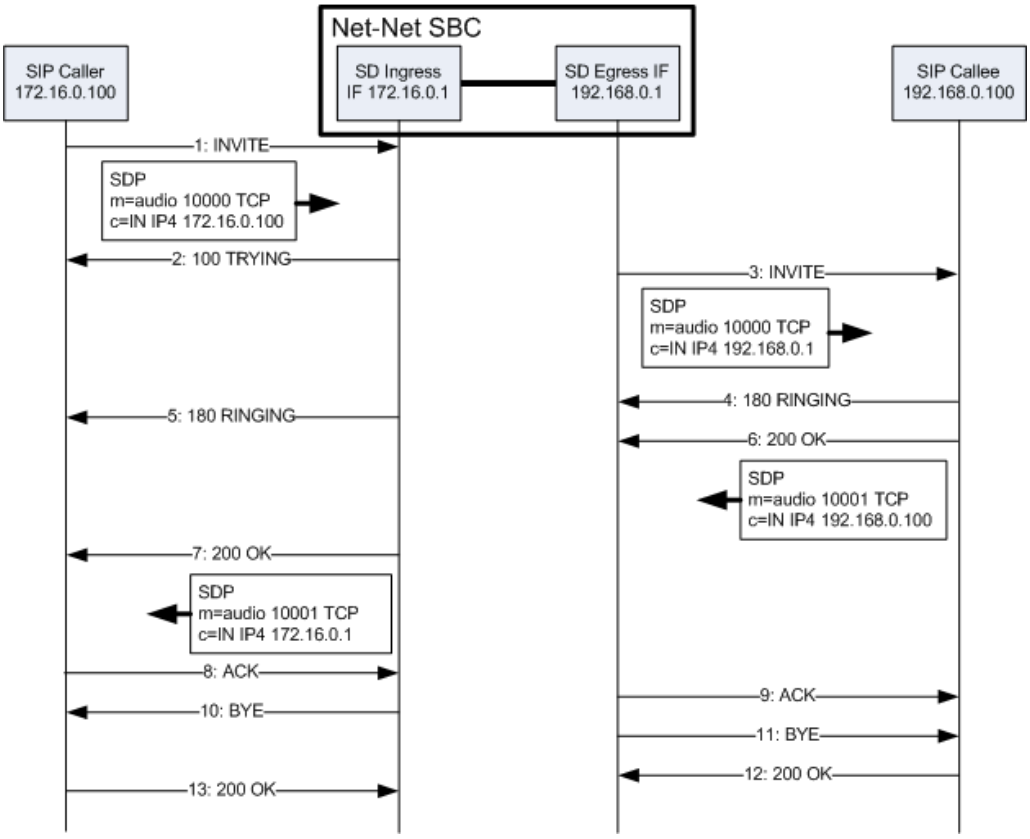


The Oracle USM can support a total of eight media-over-TCP connections per call. A maximum of 4 connections are supported as initiated from each side of the call.

SDP Offer Example

The following abbreviated call flow diagram sets up a media-over-TCP flow. Observe that the caller listens for audio over TCP on 172.16.0.10:10000, as described in the SDP offer (1). The Oracle USM re-writes the m and c lines in the SDP offer to reflect that it is listening for audio over TCP on its egress interface at 192.168.0.1:10000 (3). The Oracle USM then forwards the SIP invite to the callee.

The SIP callee responds with an SDP answer in a 200 OK message. The callee indicates it is listening for the audio over TCP media on 192.168.0.10:10001 (6). The Oracle USM re-writes the m and c lines in the SDP answer to reflect that it is listening for audio over TCP on the call's ingress interface at 172.16.0.1:10001 (7). The Oracle USM then forwards the SIP invite to the caller.



All interfaces involved with the end-to-end TCP flow have now established their listening IP address and port pairs.

Timers

The Oracle USM has three guard timers that ensure a TCP media flow does not remain connected longer than configured. You can set each of these from 0 (disabled) to 999999999 in seconds.

- TCP initial guard timer — Sets the maximum time in seconds allowed to elapse between the initial SYN packet and the next packet in this flow.
- TCP subsequent guard timer — Sets the maximum time in seconds allowed to elapse between all subsequent sequential TCP packets.
- TCP flow time limit — Sets the maximum time that a single TCP flow can last. This does not refer to the entire call.

TCP Port Configuration

To configure media over TCP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring media over TCP.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **tcp-number-of-ports-per-flow—Enter the number of ports, inclusive of the server port, to use for media over TCP. The total number of supported flows is this value minus one. The default is 2. The valid range is:**
   - Minimum—2
   - Maximum—5

```
ORACLE(realm-config)# tcp-number-of-ports-per-flow 5
```

5. **tcp-initial-guard-timer—Enter the maximum time in seconds allowed to elapse between the initial SYN packet and the next packet in a media-over-TCP flow. The default is 300. The valid range is:**
   - Minimum—0
   - Maximum—999999999

```
ORACLE(realm-config)# tcp-initial-guard-timer 300
```

6. **tcp-subsq-guard-timer—Enter the maximum time in seconds allowed to elapse between all subsequent sequential media-over-TPC packets. The default is 300.**
   - Minimum—0
   - Maximum—999999999

```
ORACLE(realm-config)# tcp-subsq-guard-timer 300
```

7. **tcp-flow-time-limit—Enter the maximum time in seconds that a media-over-TCP flow can last. The default is 86400. The valid range is:**
   - Minimum—0
   - Maximum—999999999

```
ORACLE(realm-config)# tcp-flow-time-limit 86400
```

Restricted Media Latching

The restricted media latching feature lets the Oracle USM latch only to media from a known source IP address, in order to learn and latch the dynamic UDP port number. The restricting IP address's origin can be either the SDP information or the SIP message's Layer 3 (L3) IP address, depending on the configuration.

About Latching

Latching is when the Oracle USM listens for the first RTP packet from any source address/port for the destination address/port of the Oracle USM. The destination address/port is allocated dynamically and sent in the SDP. After it receives a RTP packet for that allocated destination address/port, the Oracle USM only allows subsequent RTP packets from that same source address/port for that particular Oracle USM destination address/port. Latching does not imply that the latched source address/port is used for the destination of the reverse direction RTP packet flow (it does not imply the Oracle USM will perform symmetric RTP).

Restricted Latching

The Oracle USM restricts latching of RTP/RTCP media for all calls within a realm. It latches to media based on one of the following:

- SDP: the IP address and address range based on the received SDP c= connect address line in the offer and answer.
- Layer 3: the IP address and address range based on the received L3 IP address of the offer or answer. This option is for access registered HNT endpoints. If the L3 IP address is locally known and cached by the Oracle USM as the public SIP contact address, that information could be used instead of waiting for a response. The Oracle USM might use the L3 IP address restriction method for all calls regardless of whether the endpoint is behind a NAT or not, for the same realms.

Symmetric Latching

A mode where a device's source address/ports for the RTP/RTCP it sends to the Oracle USM (USM) that are latched, are then used for the destination of RTP/RTCP sent to the device.

After allocating the media session in SIP, the USM sets the restriction mode and the restriction mask for the calling side as well as for the called side. It sets the source address and address prefix bits in the flow. It also parses and loads the source flow address into the MIBOCO messages. After receiving the calling SDP, the USM sets the source address (address and address prefix) in the appropriate flow (the flow going from calling side to the called side). After receiving the SDP from the called side, the USM sets the source address in the flow going from the called side to the calling side.

The USM uses either the address provided in the SDP or the layer 3 signaling address for latching. You also configure the USM to enable latching so that when it receives the source flow address, it sets the address and prefix in the NAT flow. When the NAT entry is installed, all the values are set correctly. In addition, sipd sends the information for both the incoming and outgoing flows. After receiving SDP from the called side sipd, the USM sends information for both flows to the MBCD so that the correct NAT entries are installed.

Enabling restricted latching may make the USM wait for a SIP/SDP response before latching, if the answerer is in a restricted latching realm. This is necessary because the USM does not usually know what to restrict latching to until the media endpoint is reached. The only exception could be when the endpoint's contact/IP is cached.

Relationship to Symmetric Latching

The current forced HNT symmetric latching feature lets the Oracle USM assume devices are behind NATs, regardless of their signaled IP/SIP/SDP layer addresses. The Oracle USM latches on any received RTP destined for the specific IP address/port of the Oracle USM for the call, and uses the latched source address/port for the reverse flow destination information.

If both restricted latching and symmetric latching are enabled, the Oracle USM only latches if the source matches the restriction, and the reverse flow will only go to the address/port latched to, and thus the reverse flow will only go to an address of the same restriction.

- Symmetric latching is enabled.
  If symmetric latching is enabled, the Oracle USM sends the media in the opposite direction to the same IP and port, after it latches to the source address of the media packet.
- Symmetric latching is disabled.

If symmetric latching is disabled, the Oracle USM only latches the incoming source. The destination of the media in the reverse direction is controlled by the SDP address.

Example 1

A typical example is when the Oracle USM performs HNT and non-HNT registration access for endpoints. Possibly the SDP might not be correct, specifically if the device is behind a NAT. Therefore the Oracle USM needs to learn the address for which to restrict the media latching, based on the L3 IP address. If the endpoint is not behind a NAT, then the SDP could be used instead if preferred. However, one can make some assumptions that access-type cases will require registration caching, and the cached fixed contact (the public FW address) could be used instead of waiting for any SDP response.

Example 2

Another example is when a VoIP service is provided using symmetric-latching. A B2BUA/proxy sits between HNT endpoints and the Oracle USM, and calls do not appear to be behind NATs from the Oracle USM's perspective. The Oracle USM's primary role, other than securing softswitches and media gateways, is to provide symmetric latching so that HNT media will work from the endpoints.

To ensure the Oracle USM's latching mechanism is restricted to the media from the endpoints when the SIP Via and Contact headers are the B2BUA/proxy addresses and not the endpoints', the endpoint's real (public) IP address in the SDP of the offer/answer is used. The B2BUA/proxy corrects the c= line of SDP to that of the endpoints' public FW address.

The Oracle USM would then restrict the latching to the address in the SDP of the offer from the access realm (for inbound calls) or the SDP answer (for outbound calls).

Restricted Latching Configuration

To configure restricted latching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to apply this feature.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: Acme_Realm <none>           0.0.0.0
   2: H323REALM  <none>           0.0.0.0
   selection:1
   ORACLE(realm-config)#
   ```

5. **restricted-latching**— Enter the restricted latching mode. The default is **none**. The valid values are:

- **none**—No restricted-latching used
- **sdp**—Use the address provided in the SDP for latching
- **peer-ip**—Use the layer 3 signaling address for latching

6. **restriction-mask**— Enter the number of address bits you want used for the source latched address. This field will be used only if the restricted-latching is used. The default is **32**. When this value is used, the complete IP address is matched for IPv4 addresses. The valid range is:
   - Minimum—1
   - Maximum—128

Media Release Across SIP Network Interfaces

This feature lets the Oracle USM release media between two SIP peers, between two realms on two network interfaces of the same Oracle USM. Use this feature when you want the Oracle USM to release media for specific call flows, regardless of the attached media topology.

Media Release Configuration

To configure media release across network interfaces:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to apply this feature.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: Acme_Realm <none>           0.0.0.0
   selection:1
   ORACLE(realm-config)#
   ```

5. **mm-in-system**—Set this parameter to **enabled** to manage/latch/steer media in the Oracle USM. Set this parameter to **disabled** to release media in the Oracle USM.

   > **Note:**
   >
   > Setting this parameter to disabled will cause the Oracle USM to NOT steer media through the system (no media flowing through this Oracle USM).

   The default is **enabled**. The valid values are:

   - enabled | disabled

Media Release Behind the Same IP Address

The media management behind the same IP feature lets the Oracle USM release media when two endpoints are behind the same IP address, in the same realm. Using this feature prevents the media for intra-site calls from going through the Oracle USM. You can use this feature for both hosted NAT traversal (HNT) and non-HNT clients. It works with NATed endpoints and for non-NATed ones that are behind the same IP.

Additional Media Management Options

Additional media management options include:

- Media directed between sources and destinations within this realm on this specific Oracle USM. Media travels through the Oracle USM rather than straight between the endpoints.
- Media directed through the Oracle USM between endpoints that are in different realms, but share the same subnet.
- For SIP only, media released between multiple Oracle USMs.
  To enable SIP distributed media release, you must set the appropriate parameter in the realm configuration. You must also set the SIP options parameter to media-release with the appropriate header name and header parameter information. This option defines how the Oracle USM encodes IPv4 address and port information for media streams described by, for example, SDP.

Configuring Media Release Behind the Same IP Address

You need to configure both the mm-in-realm and mm-same-ip parameters for the realm:

- If the mm-in-realm parameter is disabled, the mm-same-ip parameter is ignored.
- If the mm-in-realm parameter is enabled and the mm-same-ip parameter is disabled, media will be managed in the realm but released if the two endpoints are behind the same IP address.

To configure media management:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **mm-in-realm**—Enable if you plan to use **mm-same-ip**. If this parameter is disabled, the **mm-same-ip** parameter is ignored. If you set this to **enabled** and mm-same-ip to **disabled**, media is managed in the realm but released if the two endpoints are behind the same IP address. The default is **disabled**. The valid values are:

- enabled | disabled

5. **mm-same-ip**—Enable if you want media to go through this Oracle USM, if **mm-in-realm** is **enabled**. When **disabled**, the media will not go through the Oracle USM for endpoint that are behind the same IP. The default is **enabled**. The valid values are:
   - enabled | disabled

Bandwidth CAC for Media Release

The bandwidth CAC for media release feature adds per-realm configuration that determines whether or not to include inter-realm calls in bandwidth calculations. When you use this feature, the Oracle USM's behavior is to count and subtract bandwidth from the used bandwidth for a realm when a call within a single site has its media released. When you do not enable this feature (and the Oracle USM's previous behavior), the Oracle USM does not subtract the amount of bandwidth.

In other words:

- When you enable this feature, an inter-realm media-released call will decrement the maximum bandwidth allowed in that realm with the bandwidth used for that call.
- When you disable this feature (default behavior), and inter-realm media-released call will not decrement the maximum bandwidth allowed for that call.

Bandwidth CAC Configuration

To enable bandwidth CAC for media release:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to want to add this feature.

   ```
   ORACLE(realm-config)# select
   ```

5. **bw-cac-non-mm**—Enable this parameter to turn on bandwidth CAC for media release. The default is **disabled**. The valid values are:
   - enabled | disabled

6. Save and activate your configuration.

Media Release between Endpoints with the Same IP Address

You can configure your Oracle USM to release media between two endpoints even when one of them:

- Is directly addressable at the same IP address as a NAT device, but is not behind a NAT device
- Is at the same IP address of a NAT device the other endpoint is behind

You enable this feature on a per-realm basis by setting an option in the realm configuration.

When this option is not set, theOracle USM will (when configured to do so) release media between two endpoints sharing one NAT IP address in the same realm or network.

Media Release Configuration

In order for this feature to work properly, the following conditions apply for the realm configuration:

- Either the **mm-in-realm** or the **mm-in-network** parameter must be disabled; you can have one of these enabled as long as the other is not. The new option will apply to the parameter that is disabled.
- If either the **mm-in-realm** or **mm-in-network** parameter is enabled, then the **mm-same-ip** parameter must be disabled.

To enable media release between endpoints with the same IP address:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **release-media-at-same-nat** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(realm-config)# options +release-media-at-same-nat
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Media Release Behind the Same NAT IP Address

You can now configure your Oracle USM to release media between endpoints sharing the same NAT IP address, even if one endpoint is at—but not behind—the same NAT. This feature expands on the Oracle USM'S pre-existing ability to release media between calling and called parties behind the same IP address/NAT device in the same realm or network.

Media Release Configuration

For this feature to work properly, your realm configuration should either have the **mm-in-realm** or **mm-in-network** parameter set to disabled, unless the **mm-same-ip** parameter is set to disabled. If the **mm-same-ip** parameter is enabled, then **mm-in-realm** or **mm-in-network** can both be enabled.

To set the option that enables media release behind the same IP address:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **release-media-at-same-nat** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(realm-config)# options +release-media-at-same-nat
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Codec Reordering

Certain carriers deploy voice services where their peering partners do not use the carriers' preferred codecs. The Oracle USM can now reorder the codecs so that the preferred one is selected first.

Take the example of a carrier that deploys a voice service using G.729 rather than G.711. If that carrier has a peering partner providing call origination for the VoIP customers with G.711 used as the preferred codec, there can be issues with codec selection.

The Oracle USM resolves this issue by offering its codec reordering feature. Enabled for realms and session agents, this feature gives the Oracle USM the ability to reorder the default codec in an SDP offer to the preferred codec before it forwards the offer to the target endpoint. When you enable this feature, you increase the probability that the target endpoint will choose the preferred codec for its SDP answer, thereby avoiding use of the undesired codec.

You enable codec reordering feature by setting the preferred-codec=X (where X is the preferred codec) option in the realm and session agent configurations. You set it in the realm from which the Oracle USM receives SDP offers (in requests or responses), and for which the media format list needs to be reordered by the Oracle USM prior to being forwarded. To configure additional

codec ordering support for cases when a response or request with an SDP offer is from a session agent, you can set this option in the session agent configuration.

If you enable the option, the Oracle USM examines each SDP media description before if forwards an SDP offer. And if necessary, it performs reordering of the media format list to designate that the preferred codec as the default.

The Oracle USM determines preferred codecs in the following ways:

- If the response or request with an SDP offer is from a session agent, the Oracle USM determines the preferred codec by referring to the session agent configuration. You set the preferred codec for a session agent by configuring it with the preferred-codec=X option.
- If the response or request with an SDP offer is not from a session agent or is from a session agent that does not have the preferred-codec=X option configured, the Oracle USM determines the preferred codec by referring to the preferred-codec=X option in the realm.
- If the Oracle USM cannot determine a preferred codec, it does not perform codec reordering.

The way that the Oracle USM performs codec reordering is to search for the preferred codec in the SDP offer's media description (m=) line, and designate it as the default codec (if it is not the default already). After it marks the preferred codec as the default, the Oracle USM does not perform any operation on the remaining codecs in the media format list.

> **Note:**
>
> that the Oracle USM performs codec reordering on the media format list only. If the rtpmap attribute of the preferred codec is present, the Oracle USM does not reorder it.

Preferred Codec Precedence

When you configure preferred codecs in session agents or realms, be aware that the codec you set for a session agent takes precedence over one you set for a realm. This means that if you set preferred codecs in both configurations, the one you set for the session agent will be used.

In the case where the Oracle USM does not find the session agent's preferred codec in the SDP offer's media format list, then it does not perform codec reordering even if the media format list contains the realm's preferred codec.

Codec Reordering Configuration

When you configure codec ordering, the codec you set in either the session agent or realm configuration must match the name of a media profile configuration. If your configuration does not use media profiles, then the name of the preferred codec that you set must be one of the following:

- PCMU
- G726-32
- G723
- PCMA
- G722
- G728

- G729

> **Note:**
>
> If you configure this feature for a session agent, you must configure it for the associated realm as well. Otherwise, the feature will not work correctly.

Setting a Preferred Codec for a Realm

To set a preferred codec for a realm configuration:

These instructions assume that you want to add this feature to a realm that has already been configured.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to apply this feature.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: public     media2:0         0.0.0.0
   2: private    media1:0         0.0.0.0
   selection:1
   ORACLE(realm-config)#
   ```

5. **options**—Set the **options** parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**preferred-codec=X**), and then press Enter. X is the codec that you want to set as the preferred codec.

   ```
   ORACLE(realm-config)# options +preferred-codec=PCMU
   ```

   If you type options preferred-codec=X, you will overwrite any previously configured options. In order to append the new option to the realm-config's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

Setting a Preferred Codec for a Session Agent

To set a preferred codec for a session agent configuration:

These instructions assume that you want to add this feature to a session agent that has already been configured.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. Select the session agent where you want to apply this feature.

```
ORACLE(session-agent)# select
<hostname>:
1: acmepacket.com realm=       ip=
2: sessionAgent2  realm=tester ip=172.30.1.150
selection:
selection:1
ORACLE(session-agent)#
```

5. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**preferred-codec=X**), and then press Enter. X is the codec that you want to set as the preferred codec.

```
ORACLE(session-agent)# options +preferred-codec=PCMU
```

If you type options preferred-codec=X, you will overwrite any previously configured options. In order to append the new option to the session agent's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

6. Save and activate your configuration.

Media Profiles Per Realm

For different codecs and media types, you can set up customized media profiles that serve the following purposes:

- Police media values
- Define media bandwidth policies

You can use media policies globally for the Oracle USM, or you can configure them for application on a per-realm basis. For a realm, you can configure a list of media profiles you want applied. The Oracle USMmatches the value you set for the **match-media-profiles** parameter, and then applies those media profiles to the realm itself and to all of its child realms (but not to its parent realms).

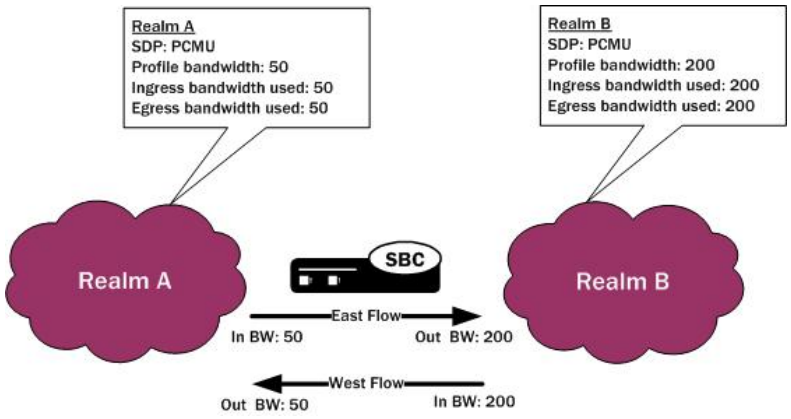
> **Note:**
>
> This feature has no impact on the ways the Oracle USM uses media profiles non-realm applications such as: SIP interfaces, session agents, codec policies, and policy attributes.

Call Admission Control and Policing

The Oracle USM supports call admission control (CAC) based on realm, and it applies the limits on either ingress or egress bandwidth counters. If a calls exceeds bandwidth on either the

ingress or egress side, the Oracle USM rejects the call. You can also use per-user CAC, which limits the maximum bandwidth from the east and west flows for both the TO and FROM users.

When you apply media profiles to a realm, the Oracle USM applies bandwidth policing from the flow's ingress realm media profile. In the diagram below, the Oracle USM policies traffic for Realm A based Realm A's policing values, and the same is true for Realm B.



Media Profile Configuration

This section shows you how to configure multiple media profiles per realm, and it explains how to use wildcarding.

To reference a media profile in this list, you need to enter its name and subname values in the following format <name>::<subname>. Releases C6.1.0 and later accept the subname so you can configure multiple media profile for the same codec; the codec **name** customarily serves and the name value for a media profile configuration.

About Wildcarding

You can wildcard both portions (name and subname) of this value:

- When you wildcard the **name** portion of the value, you can provide a specific subname that the Oracle USM uses to find matching media profiles.
- When you wildcard the subname portion of the value, you can provide a specific **name** that the Oracle USM uses to find matching media profiles.

You can also enter the name value on its own, or wildcard the entire value. Leaving the subname value empty is also significant in that it allows the realm to use all media profile that have no specified **subname**. However, you cannot leave the **name** portion of the value unspecified (as all media profiles are required to have names).

Consider the examples in the following table:

Syntax	Example Value	Description
<name>	PCMU	Matches any and all media profiles with the name value configured as PCMU. This entry has the same meaning as a value with this syntax: <name>::*.

Syntax	Example Value	Description
<name>::	PCMU::	Matches a media profile with the name with the name value configured as PCMU with an empty subname parameter.
<name>::<subname>	PCMU::64k	Matches a media profiles with the name with the name value configured as PCMU with the subname parameter set to 64k.
*	*	Matches anything, but does not have to be a defined media profile.
::	*::*	Matches any and all media profiles, but requires the presence of media profile configurations.
*::<subname>	*::64k	Matches all media profiles with this subname. You might have a group of media profiles with different names, but the same subname value.
*::	*::	Matches any media profiles with an empty subname parameter.
::	::	Invalid
::*	::*	Invalid

The Oracle USM performs matching for wildcarded **match-media-profiles** values last. Specific entries are applies first and take precedence. When the Oracle USM must decide between media profiles matches, it selects the first match.

To use media profiles for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **match-media-profiles**—In the form **<name>::<subname>**, enter the media profiles you would like applied to this realm. These values correspond to the name and subname parameters in the media profile configuration. You can wildcard either of these portions of the value, or you can leave the <subname> portion empty.

   This parameter has no default.

5. Save and activate your configuration.

Multiple Media Profiles

You can use the media profiles configuration to set up:

- One media profile for a particular SIP SDP encoding (such as G729), where the name of the profile identifies it uniquely. This behavior is your only option in Oracle USM release prior to Release C6.1.0.

- Multiple media profiles for the same SIP SDP encoding. Available in Release C6.1.0 and forward, you can create multiple media profiles for the same encoding. To do so, you add a subname to the configuration, thereby identifying it uniquely using two pieces of information rather than one.

The sections below provide two descriptions of deployments where using multiple profiles for the same codec would solve codec and packetization problems for service providers.

Use Case 1

Service Provider 1 peers with various carriers, each of which uses different packetization rates for the same codec. For example, their Peer 1 uses 10 milliseconds G.711 whereas their Peer 2 uses 30 milliseconds for the same codec. The difference in rates produces a difference in bandwidth consumption—resulting in a difference in SLA agreements and in Oracle USM call admission control (CAC) and bandwidth policing. Service Provider 1 uses the Oracle USM's media profile configuration parameters to determine CAC (**req-bandwidth**) and bandwidth policing (**avg-rate-limit**). Because this service provider's peers either do not use the SDP p-time attribute or use it inconsistently, it is difficult to account for bandwidth use. And so it is likewise difficult to set up meaningful media profiles.

The best solution for this service provider—given its traffic engineering and desire for the cleanest routing and provisioning structures possible—is to define multiple media profiles for the same codec.

Use Case 2

Service Provider 2 supports H.263 video, for which the Oracle USM offers a pre-provisioned media profile with a set bandwidth value. And yet, H.263 is not a codec that has a single bandwidth value. Instead, H.263 can have different bandwidth values that correspond to various screen resolution and quality. While it is true that the Oracle USMcan learn the requisite bandwidth value from SDP, not all SDP carries the bandwidth value nor do system operators always trust the values communicated.

Configuring multiple media profiles for the same codec (here, H.263) helps considerably with this problem—and moves closer to complete solution. Service Provider 2 can configure H.263 media profiles capable of handling the different bandwidth values that might appear.

Multiple Media Profiles Configuration

Configuring the **subname** parameter in the media profiles configuration allows you to create multiple media profiles with the same name.

To configure the subname parameter for a media profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter. If you are adding this feature to a pre-existing media profile configuration, you will need to select and edit your media profile.

```
ORACLE(session-router)# media-profile
ORACLE(media-profile)#
```

4. **subname**—Enter the subname value for this media profile. Information such as the rate or bandwidth value make convenient subname values. For example, you might set the **name** of the media profile as PCMU and the **subname** as 64k.

   This parameter is not require and has no default.

5. Save and activate your configuration.

SIP Disable Media Inactivity Timer for Calls Placed on Hold

Hardware-based media flow guard timers detect when a call has lost media while it is being relayed through the Oracle USM. In response, the system tears down the call.

You can configure **disable-guard-timer-sendonly** to disable media inactivity timers for calls placed on hold. The Oracle USM disableds initial and subsequent guard timers for media when the SIP or IWF call is put on hold with a 0.0.0.0 address in:

- The c=connection line
- An a=inactive attribute
- An a=sendonly attribute

It should be noted that disabling the media inactivity timers will also disable the guard timers for calls which are not necessarily on hold, but simply are one-way audio applications.

No license requirements to enable this feature.

Media Inactivity Timer Configuration

To disable the media inactivity timer for calls placed on hold:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager-config** and press Enter.

   ```
   ORACLE(media-manager)# media-manager-config
   ORACLE(media-manager-config)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **disable-guard-timer-sendonly** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(media-manager-config)# options +disable-guard-timer-sendonly
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Peer-to-Peer MSRP TCP Stitching

The Oracle USM supports peer-to-peer TCP connections for peers behind NATs, enabling Message Session Relay Protocol (MSRP) client to communicate with one another. More specifically, the Oracle USM can:

- Establish incoming TCP connections with each endpoint participating in the MSRP session using a 3-way handshake. The Oracle USM receives incoming SYNs on the local address and port provided in the SDP offer and answer to each endpoint.
- Stitch together the two TCP connections internally after successful establishment of both connections. This capability is used when the caller and the callee initiate TCP SYNs towards one another via the Oracle USM; the stitching makes both clients think they are talking to a server. To achieve this end, the Oracle USM caches SYNs from both sides so it can modify the SYN packets to SYN-Acks with the correct sequence and Ack numbers. Note, though this case is rare, that if a user is behind a NAT offers a=passive, then this feature cannot function properly.
- Relay MSRP stream between the endpoints.
- Police bandwidth for MSRP streams based on a defined media profile for MSRP.

5

Oracle USM Supporting the IMS Core

General Description

The Oracle USM functions in an IMS core. It communicates with the HSS to obtain Authorization, Authentication, S-CSCF assignment, and ultimately routing instructions. To accomplish these functions, the Oracle USM can perform the SIP registrar role in conjunction with an HSS.

Message Authentication for SIP Requests

The Oracle USM authenticates requests by configuring the sip authentication profile configuration element. The name of this configuration element is either configured as a parameter in the sip registrar configuration element's authentication profile parameter or in the sip interface configuration element's sip-authentication-profile parameter. This means that the Oracle USM can perform SIP digest authentication either globally, per domain of the Request URI or as received on a SIP interface.

After naming a sip authentication profile, the received methods that trigger digest authentication are configured in the methods parameter. You can also define which anonymous endpoints are subject to authentication based on the request method they send to the Oracle USM by configuring in the anonymous-methods parameter. Consider the following three scenarios:

- By configuring the methods parameter with REGISTER and leaving the anonymous-methods parameter blank, the Oracle USM authenticates only REGISTER request messages, all other requests are unauthenticated.
- By configuring the methods parameter with REGISTER and INVITE, and leaving the anonymous-methods parameter blank, the Oracle USM authenticates all REGISTER and INVITE request messages from both registered and anonymous endpoints, all other requests are unauthenticated.
- By configuring the methods parameter with REGISTER and configuring the anonymous-methods parameter with INVITE, the Oracle USM authenticates REGISTER request messages from all endpoints, while INVITES are only authenticated from anonymous endpoints.

User Authorization

In an IMS network, the Oracle USM requests user authorization from an HSS when receiving a REGISTER message. An HSS is defined on the Oracle USM by creating a home subscriber server configuration element that includes a name, ip address, port, and realm as its basic defining data.

UAR/UAA Transaction

Before requesting authentication information, the Oracle USM sends a User Authorization Request (UAR) to the HSS for the registering endpoint to determine if this user is allowed to receive service. The Oracle USM populates the UAR's AVPs as follows:

- Public-User-Identity—the SIP AOR of the registering endpoint
- Visited-Network-Identity—the value of the network-id parameter from the ingress sip-interface.
- Private-User-Identity—the username from the SIP authorization header, if it is present. If not, this value is the public User ID.
- User-Authorization-Type—always set to REGISTRATION_AND_CAPABILITIES (2)

The Oracle USM expects the UAA to be either:

- DIAMETER_FIRST_REGISTRATION
- DIAMETER_SUBSEQUENT_REGISTRATION

Any of these responses result in the continued processing of the registering endpoint. Any other result code results in an error and a 403 returned to the registering UA (often referred to as a UE). The next step is the authentication and request for the H(A1) hash.

SIP Digest User Authentication

Authentication via MAR/MAA

To authenticate the registering user, the Oracle USM needs a digest realm, QoP, and the H(A1) hash. It requests these from a server, usually the HSS, by sending it a Multimedia Auth Request (MAR) message. The MAR's AVPs are populated with:

- Public-User-Identity—the SIP AOR of the endpoint being registered (same as UAR)
- Private-User-Identity—the username from the SIP authorization header or the SIP AOR if the AOR for PUID parameter is enabled. (Same as UAR)
- SIP-Number-Auth-Items—always set to 1
- SIP-Auth-Data-Item -> SIP-Item-Number—always set to 1
- SIP-Auth-Data-Item -> SIP-Authentication-Scheme—always set to SIP_DIGEST
- Server-Name—the home-server-route parameter in the sip registrar configuration element. It is the URI (containing FQDN or IP address) used to identify and route to this Oracle USM.

The Oracle USM expects the MAA to include a SIP-Auth-Data-Item VSA, which includes digest realm, QoP and H(A1) information as defined in RFC2617. The information is cached for subsequent requests. Any result code received from the HSS other than DIAMETER_SUCCESS results in a 403 error response returned for the original request.

The MAR/MAA transaction is conducted with the server defined in the credential retrieval config parameter found in the sip-authentication profile configuration element. This parameter is populated with the name of a home-subscriber-server configuration element.

SIP Authentication Challenge

When the Oracle USM receives a response from the HSS including the hash value for the user, it sends a SIP authentication challenge to the endpoint, if the endpoint did not provide any authentication headers in its initial contact the with Oracle USM. If the endpoint is registering, the Oracle USM replies with a 401 Unauthorized message with the following WWW-Authenticate header:

```
WWW-Authenticate: Digest realm="atlanta.com", domain="sip:boxesbybob.com",
qop="auth", nonce="f84f1cec41e6cbe5aea9c8e88d359", opaque="", stale=FALSE,
algorithm=MD5
```

If the endpoint initiates any other request to the Oracle USM besides REGISTER, the Oracle USM replies with a 407 Proxy Authentication Required message with the following Proxy-Authenticate header:

```
Proxy-Authenticate: Digest realm="atlanta.com", qop="auth",
nonce="f84f1cec41e6cbe5aea9c8e88d359", opaque="", stale=FALSE, algorithm=MD5
```

Authentication Header Elements

- Domain—A quoted, space-separated list of URIs that defines the protection space. This is an optional parameter for the "WWW-Authenticate" header.
- Nonce—A unique string generated each time a 401/407 response is sent.
- Qop—A mandatory parameter that is populated with a value of "auth" indicating authentication.
- Opaque—A string of data, specified by the Oracle USM which should be returned by the client unchanged in the Authorization header of subsequent requests with URIs in the same protection space.
- Stale—A flag indicating that the previous request from the client was rejected because the nonce value was stale. This is set to true by the SD when it receives an invalid nonce but a valid digest for that nonce.
- Algorithm—The Oracle USM always sends a value of "MD5"

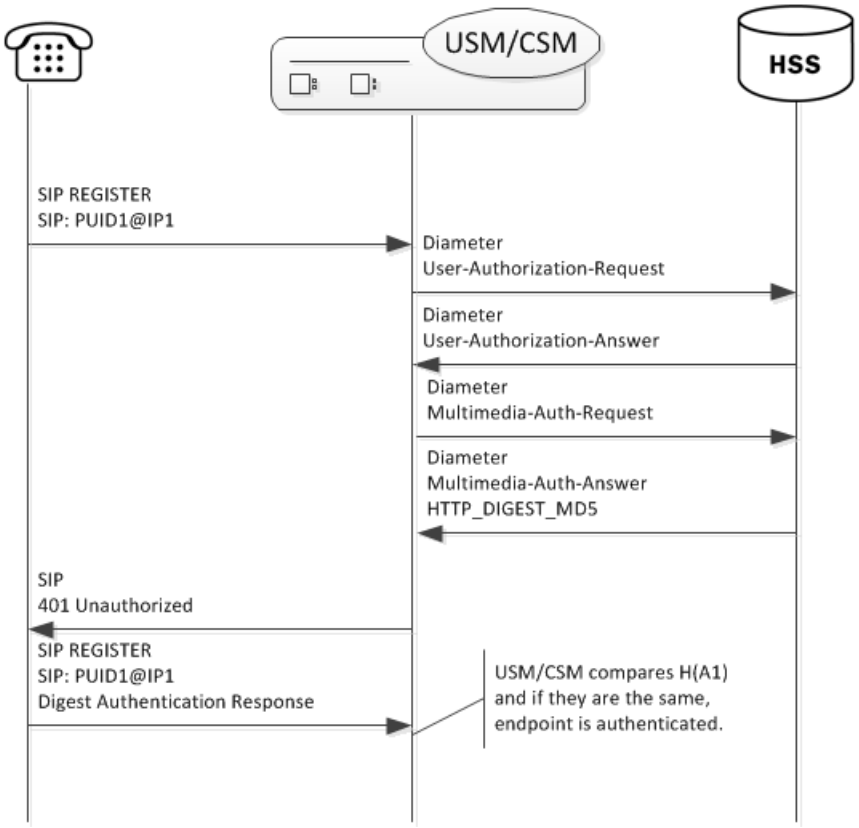
SIP Authentication Response

After receiving the 401/407 message from the Oracle USM, the UA resubmits its original request with an Authorization: header including its own internally generated MD5 hash.

Oracle USM Authentication Check

At this point, the Oracle USM has received an MD5 hash from the HSS and an MD5 hash from the UA. The Oracle USM compares the two values and if they are identical, the endpoint is successfully authenticated. Failure to match the two hash values results in a 403 or 503 sent to the authenticating endpoint.

The following image shows the User Authorization and Authentication process:



Note:

Diagram information states "USM/CSM" when the applicable content applies to both the Oracle USM and the Oracle CSM.

The Oracle USM acts as a SIP Registrar and updates an HSS with the state of its registrants.

IMS-AKA Support

The Oracle USM also supports IMS-AKA for secure authentication end-to-end between UAs in an LTE network and an IMS core. It supports IMS-AKA in compliance with 3GPP specifications TS 33-203 and TS 33-102.

The goal of IMS-AKA is to achieve mutual authentication between end station termination mechanisms, such as an IP Multimedia Services Identity Module (ISIM), and the Home Network (IMS Core). Achieving this goal requires procedures both inside and outside the core. Ultimately, IMS performs the following:

- Uses the IMPI to authenticate the home network as well as the UA;
- Manages authorization and authentication information between the HSS and the UA;
- Enables subsequent authentication via authentication vectors and sequence information at the ISIM and the HSS.

The Oracle USM authenticates registrations only. This registration authentication process is similar to SIP Digest. The process accepts REGISTER requests from UAs, conducts authorization procedures via UAR/UAA exchanges and conducts authentication procedures via MAR/MAA exchanges and challenges with the UA.

Configuration and operational support are not the same on the Oracle USM and Oracle CSM. This is because the Oracle USM can perform the P-CSCF role as well as the I-CSCF and S-CSCF roles. Applicable configuration to support IMS-AKA on the P-CSCF access interface is documented in the Security chapter of the *Oracle Communications Session Border Controller ACLI Configuration Guide*. This configuration includes defining an IMS-AKA profile, enabling the **sip-interface** for IMS-AKA and configuring the **sip-port** to use the profile.

There is no configuration required for the S-CSCF role, but there is an optional configuration that specifies how may authentication vectors it can accept from the HSS. The S-CSCF stores these authentication vectors for use during subsequent authentications. Storing vectors limits the number of times the device needs to retrieve them from the HSS. The default number of authentication vectors is three.

Authentication Sequence - Registration

UAs get service from an IMS core after registering at least one IMPU. To become registered, the UA sends REGISTER requests to the IMS core, which then attempts to authenticate the UA.

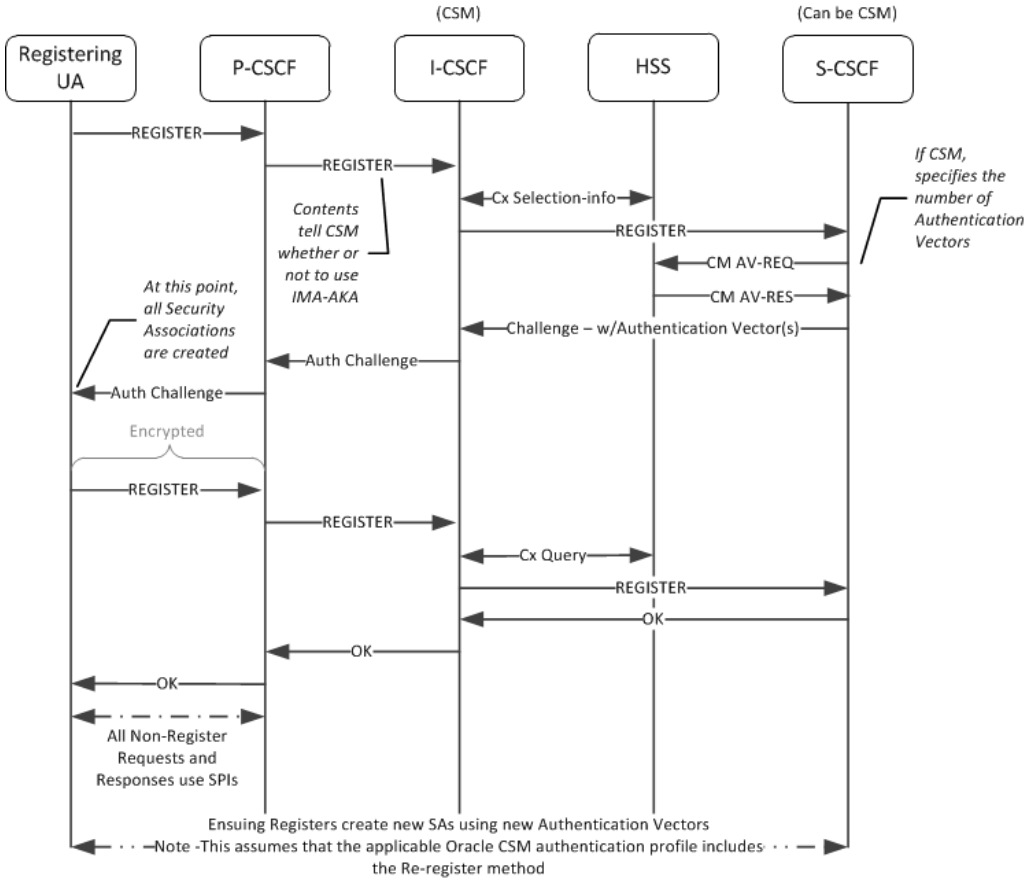
The first device to receive the REGISTER at the core is a P-CSCF, such as the Oracle USM. For the Oracle USM, appropriate configuration determines that it uses IMS-AKA as the authentication mechanism on the access interface. For an Oracle CSM, the presence and state of the “integrity-protected” parameter in the Authorization header of a REGISTER triggers the use of IMS-AKA. If the value of this parameter is either “yes” or "no", IMS-AKA is invoked. If the parameter is not present, or it is set to any other value, the Oracle USM falls back to SIP Digest authentication.

To proceed with IMS-AKA authentication, the P-CSCF engages in S-CSCF selection procedures via the I-CSCF to identify the target S-CSCF. Having identified the S-CSCF (your Oracle USM), the I-CSCF forwards the REGISTER to it. The I-CSCF next engages in standard UAR and MAR procedures. For IMS-AKA deployments, the HSS follows procedures defined in TS 33-203 to create authentication vectors for the UA. The HSS provides the vectors to the S-CSCF, which then proceeds with authentication procedures defined in TS 33-203.

After processing, the S-CSCF uses authentication vectors to challenge the UA. The UA uses the information in this challenge to, first, authenticate the Home Network. Having confirmed the network, the UA then prepares and sends its authentication information back towards the S-CSCF. The S-CSCF is then responsible for authenticating the UA. The S-CSCF sends a 200OK back to the UA upon successful authentication, allowing the UA to get service from the HN.

The Oracle USM caches the AOR’s registration and stores authentication vectors for subsequent authentications, thereby minimizing the work required by the HSS.

The overall sequence is depicted below.



Outside the Core

LTE networks include UAs that have an IP Multimedia Service Identity Module (ISIM) or equivalent. ISIMs are configured with a long-term key used to authenticate and calculate cipher keys, as well as IP Multimedia Private and Public Identities (IMPI and IMPU). The ISIM serves as the means of authenticating the home network to the UA. The UA, in turn, sends information based on it's ISIM configuration to the home network, which can then authenticate the UA.

Establishment of Security Associations (SAs) to and from the UA are the responsibility of the P-CSCF. The P-CSCF should also be capable of managing the processes when the UA is behind a NAT.

> **Note:**
>
> Within the context of IMS-AKA, only traffic between the P-CSCF and the UA is encrypted.

Authentication Success

When using IMS-AKA, successful registration of a UA consists of registering at least one IMPU and the IMPI authenticated within IMS. The UA begins this process by sending it REGISTER request to the P-CSCF properly specifying IMS-AKA authentication. IMS then performs standard procedures to identify the appropriate S-CSCF. Upon receipt of the

REGISTER, the S-CSCF checks for the presence of an authentication vector. If it is present the S-CSCF issues the authentication challenge; if not, it requests authentication vector(s) from the HSS. Note that the Oracle USM allows you to request multiple authentication vectors via configuration. The HSS provides the following components within an authentication vector:

- RAND—random number
- XRES—expected response
- CK—cipher key
- IK—integrity key
- AUTN—authentication token

The MAR provided to the S-CSCF differ from that of SIP digest authentication requests as follows:

- The SIP-Number-Auth-Items AVP specifies the number of authentication vectors, which is equal to the home-subscriber-server's `num-auth-vectors` setting.
- The SIP-Authentication-Scheme AVP specifies the authentication scheme, Digest-AKAv1-MD5.

At this point, the Oracle USM can send the authentication challenge to the UA. If multiple authentication vectors were provided by the HSS, the Oracle USM can independently authenticate the UA until the pool is exhausted. The S-CSCF stores the RAND it sends to the UA to resolve future synchronization errors, if any. No authentication vector can be used more than once. This is validated by the ISIM, using a sequence number (SQN).

When a P-CSCF receives an authentication challenge, it removes and stores the CK and the IK. The P-CSCF forward the rest of the information to the UA.

The UA is responsible for verifying the home network. Having received the AUTN from the P-CSCF, the UA derives MAC and SQN values. Verifying both of these, the UA next generates a response including a shared secret and the RAND received in the challenge. The UA also computes the CK and IK.

Upon receipt of this response, IMS provides the message to the S-CSCF, which determines that the XRES is correct. If so, it registers the IPMU and, via IMS sends the 200 OK back to the UA.

Authentication Failure

Either the UA or IMS can deny authentication via IMS-AKA. In the case of the UA, this is considered a network failure; in the case of IMS there would be a user authentication failure.

Network Authentication Failure

The UA determines that the HN has failed authentication, it sends a REGISTER request with an empty authorization header parameter and no authentication token for synchronization (AUTS). This indicates that the MAC parameter was invalid as determined by the UA. In this case, the S-CSCF sends a 403 Forbidden message back to the UA.

User Authentication Failure

IMS-AKA determines user authentication failure as either:

- IK incorrect—If the REGISTER includes a bad IK, the P-CSCF detects this and discards the packet at the IPSEC layer. In this case, the REGISTER never reaches the S-CSCF.

- XRES incorrect—In this case, the REGISTER reaches the S-CSCF. The S-CSCF detects the incorrect XRES, the S-CSCF sends a 4xxx Auth_Failure message back to the UA via IMS.

Synchronization

Synchronization refers to authentication procedures when the (REFRESH TIMING) is found to be stale. This is not an authentication failure.

The UA may send an AUTS in response to the challenge, indicating that the authentication vector sequence is "out-of-range". Upon receipt of the AUTS, the S-CSCF sends a new authorization vector request to the HSS. The HSS checks the AUTS and, if appropriate sends a new set of authentication vectors back the the S-CSCF. Next the S-CSCF sends 401 Unauthorized back to the UA. Assuming the UA still wants to register, this would trigger a new registration procedure.

Optional IMS-AKA Configuration

The following configuration enables the Oracle USM to specify, on a per-HSS basis, the number of authentication vectors it can download per MAR. Making this setting is not required as it has a valid default entry (3).

home subscriber server

To configure the number of authentication vectors to download from a home subscriber server (HSS):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **home-subscriber-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# home-subscriber-server
   ORACLE(home-subscriber-server)#
   ```

4. **Select**—If already configured, choose the home subscriber server for which you want to set the number of authentication vectors.
5. **num-auth-vector**— [1-10] 3 default - The number of authentication vectors downloaded from HSS per MAR. The range is from 1-10 with 3 as the default.
6. Type **done** when finished.

S-CSCF Selection Based on Capabilities

Within IMS environments, the I-CSCF identifies target S-CSCF's in response to SIP traffic for which the assigned S-CSCF is not known. Enhanced selection environments can include the HSS offering mandatory and optional capabilities for a user, and the I-CSCF selecting the best S-CSCF based on capabilities the S-CSCF is best suited to support (in addition to standard criteria). The user can configure the I-CSCF resident within Oracle CSM, Oracle USM and Oracle SLRM to support this capabilities-based S-CSCF selection. Resultant operation is compliant with ETSI TS 129 228 and ETSI TS 129 229.

S-CSCF selection based on capabilities utilizes AVP information exchanged with the HSS to identify required and preferred capabilities on a per-user basis. Capabilities themselves vary widely. Examples include administrator routing preferences for divergent service types. Capabilities are manually defined at the HSS for endpoints or groups of endpoints. The Oracle CSM, Oracle USM and Oracle SLRM user configures tables on the I-CSCF that map the S-CSCF's with the capabilities they support. Further configuration enables the I-CSCF to make the best S-CSCF selection, then forward appropriately.

Diameter messaging that can generate capabilities parsing for S-CSCF selection includes UAR/UAA and LIR/LIA traffic. Inclusion of the capabilities AVPs in the message sequence triggers this enhanced S-CSCF selection by the I-CSCF.

Configuration on the HSS and the I-CSCF must be compatible in deployments that use this feature. Configuration required on the Oracle device performing the I-CSCF function includes:

- servers-capabilities-list—A sip-registrar parameter that allows you to configure the registrar with a servers-capabilities-table.
- servers-capabilities-table—A multi-instance element that names the table and includes multiple servers-capability.
  - servers-capability—A multi-instance element within the servers-capabilities-table that includes a capability (capability value associated with users and supported by servers in the list) and a server-name-list that identifies the servers that support this capability.

The USM verifies the **servers-capabilities-list** attribute with the **servers-capabilities-table** each time it loads the configuration. If the **servers-capabilities-table** with the name specified in the **servers-capabilities-list** does not exist , the system outputs the following message:

ERROR: sip-registrar [<object-name>] has invalid servers-capabilities-list entry [<entry-name>]

Server-Capabilities AVP

The Server-Capabilities AVP is a group AVP including the Mandatory-Capability AVP and Optional-Capability AVP. The number of Mandatory-Capability and Optional-Capability AVPs is not limited in a Server-Capabilities AVP. The AVP symbol notation, format and reference follows:

3GPP 32.299 states the following symbols are used in the message format definitions:

- <AVP> indicates a mandatory AVP with a fixed position in the message.
- {AVP} indicates a mandatory AVP in the message.
- [AVP] indicates an optional AVP in the message.
- *AVP indicates that multiple occurrences of an AVP is possible.

Format definitions include:

- Server-Capabilities ::= <AVP header: 603 10415>
- *{Mandatory-Capability}
- *[Optional-Capability]
- *[Server-Name] (not supported in this release)
- *[AVP] (not supported in this release)

AVP reference, including column definition and AVP table follows:

- AVP Name
- AVP Number
- Reference where the AVP was defined
- Type of data format used to express the AVP's data
- If a grouped AVP, the names of the AVPs in the group

AVP	Number	Reference	Type	Grouped
{ Server-Capabilities }	603	Base	Grouped	Mandatory-Capability Optional-Capability
{ Mandatory-Capability }	604	Base	Unsigned32	
[Optional-Capability]	605	Base	Unsigned32	

Selection Process without SLRM

The capabilities-oriented S-CSCF selection algorithm on the Oracle CSM and Oracle USM S-CSCF include selections based on mandatory and optional capabilities information received from HSS and the configured S-CSCF Capabilities Database.

The general approach to selection within this scenario include the following principles:

- Only S-CSCFs with all mandatory capabilities can be selected.
- The process gives priority to the S-CSCF with the most optional capabilities.
- The process gives priority to the local S-CSCF.
- The system attempts to spread assignments to remote S-CSCFs of the same priority.

The capabilities-oriented S-CSCF selection algorithm uses the following high-level steps within the I-CSCF function to arrive at a selection:

1. Determine that the capabilities algorithm is required:
   a. No server-name in the LIA or UAA.
   b. Capability list exists.
   c. Assigned S-CSCF flag is not set.
   d. Mandatory/Optional Capabilities received in UAA/LIA.
2. Identify potential S-CSCFs, which must support all mandatory capabilities:
   a. Ensure the S-CSCF capabilities database is configured.
   b. Build capable S-CSCF list. This list contains all S-CSCFs from the S-CSCF capabilities database that support the Mandatory capabilities.
   c. Ensure that the capable S-CSCF list is not empty. If the capable S-CSCF list is empty, return an error to the UE.
3. Ensure that the I-CSCF is not SLRM.
4. Complete capabilities selection process using optional capabilities as criteria:
   a. An S-CSCF has the most optional capabilities.
   (If so, forward.)

   b. The local S-CSCF can take on more users, has all mandatory capabilities, and has most optional capabilities.
   (If so, forward locally.)
   c. Use round robin to select the S-CSCF that has most optional capabilities.
   (If so, forward.)

5. Forward message:
   a. Forward to selected S-CSCF.
   b. Remove selected S-CSCF from capabilities list.
   c. If there is an error, for example, the SIP response requires a re-assignment, check the assigned flag.
   d. If the assigned flag is set, return to the top.
   If the assigned is not set, return to the step that checks whether the capable S-CSCF list is empty.
   e. If the capable S-CSCF list is empty, return an error to the UE.
   If the capable S-CSCF list is not empty yet, perform capabilities selection process using optional capabilities as criteria again.

Selection Process with an SLRM

The capabilities-oriented S-CSCF selection algorithm on the Oracle SLRM uses standard Oracle CSM selection criteria in addition to capabilities criteria. This criteria includes cluster configuration, S-CSCF resource utilization and SLRM synchronization.

The general approach to selection within this scenario include the following principles:

- Only Oracle CSMs with all mandatory capabilities can be selected.
- The process gives priority to the Oracle CSMs in the cluster with the most optional capabilities, and is best able to take on new users.

The capabilities-oriented S-CSCF selection algorithm uses the following high-level steps, including the SLRM's selection steps, within the I-CSCF function to arrive at a selection:

1. Determine that the capabilities algorithm is required:
   a. No server-name in the LIA or UAA.
   b. Capability list exists.
   c. Assigned S-CSCF flag is not set.
   d. Mandatory/Optional Capabilities received in UAA/LIA.
2. Execute capabilities selection:
   a. Ensure the S-CSCF capabilities database is configured.
   b. Build capable S-CSCF list. This list contains all S-CSCFs from the S-CSCF capability database that support the Mandatory capabilities.
   c. Ensure that the capable S-CSCF list is not empty. If the capable S-CSCF list is empty, return an error to the UE.
3. Execute SLRM's selection procedure, cycle through all Oracle CSMs in the cluster:
   a. Identify applicable cluster. Begin to cycle through cluster.
   b. Determine whether Oracle CSM is in capable list.

c. Determine whether Oracle CSM is at 100% utilization.

d. Determine whether the next Oracle CSM support more optional capabilities.

e. Determine whether the selected Oracle CSM is synchronized.

f. Determine whether the next Oracle CSM using fewer resources.

4. Complete capabilities selection process using optional capabilities as criteria:

   a. An S-CSCF has the most optional capabilities.
   (If so, forward message.)

   b. The local S-CSCF can take on more users and has all mandatory capabilities and most optional capabilities.
   (If so, forward message locally.)

   c. Use round robin to select the S-CSCF that has most optional capabilities.
   (If so, forward message.)

5. Forward message:

   a. Forward to selected S-CSCF.

   b. Remove selected S-CSCF from capabilities list.

   c. If there is an error, for example, the SIP response requires a re-assignment, check the assigned flag.

   d. If the assigned flag is set, return to the top.
   If the assigned is not set, return to the step that checks whether the capable S-CSCF list is empty.

   e. If the capable S-CSCF list is empty, return an error to the UE.
   If the capable S-CSCF list is not empty yet, perform SLRM's selection procedure again.

ACLI Instructions

Configuring the server-capabilities-table

A **server-capabilities-table** is a multi-instance element that allows the user to name a **servers-capability** object and apply it to a **registrar**. A **servers-capability** object is a **server-capabilities-table** sub-element that includes a **capability** and multiple server names, which support that capability.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **server-capabilities-table** and press Enter to access the path.

   ```
   ORACLE(session-router)# server-capabilities-table
   ORACLE(server-capabilities-table)#
   ```

4. Enter a contiguous string to the **name** field. This name is the reference used in the registrar configuration to specify the use of this server capabilities table.

5. Type **servers-capability** and press Enter to access the path.

```
ORACLE(server-capabilities-table)# servers-capability
ORACLE(servers-capability)#
```

6. Enter a number to specify the capability **capability**. Valid entries range from 0 to 999999999.
7. Enter the names of the servers that belong to this **server-name-list**. Name format is the same as that used within the registrar's **home-server-route** field. The format is the URI (containing FQDN or IP address) used to identify a server to the HSS. Each entry in the list is enclosed with quotes and separated by comma.
8. Type **done** and **exit** twice to complete configuration of this **server-capabilities-table** configuration element.

Configuring the server-capabilities-list

To assign a server capabilities list to a sip-registrar:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter to access the session router path.

   ```
   ORACLE(session-router)#sip-registrar
   ORACLE(sip-registrar)#
   ```

4. Type **server-capabilities-list** and press Enter. Add a capability with associated servers.

   ```
   ORACLE(sip-registrar)# server-capabilities-list my_capability_list1
   ORACLE(sip-registrar)#
   ```

5. Type **done** and **exit** to complete configuration of this **sip-registrar** configuration element.

Oracle USM as Registrar

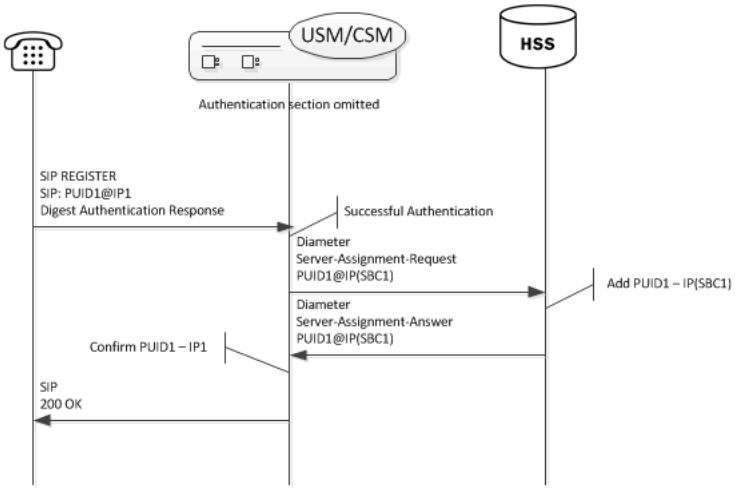
Creating a sip registrar configuration element enables the Oracle USM to act as a SIP registrar. When registration functionality is enabled, the Oracle USM actually registers endpoints rather than only caching and forwarding registrations to another device. Oracle USM registry services are enabled globally per domain, not on individual SIP interfaces or other remote logical entities.

On receiving a REGISTER message, the Oracle USM checks if it is responsible for the domain contained in the Request-URI as defined by the domains parameter and finds the corresponding sip registrar configuration. This is a global parameter—all messages are checked against all sip registrar domains. Thus you could create one sip registrar configuration element to handle all .com domains and one sip registrar configuration element to handle all .org domains. The Oracle USM begins registrar functions for all requests that match the configured domain per sip-registrar configuration element.

A UA is considered registered once a SAA assignment is received from the HSS, after which the Oracle USM sends a 200 OK message back to the registering UA.

New Registration

The following image shows a simplified call flow for a registering user:



Registration Response with the Authentication-info Header

The Oracle USM can include the authentication-info header, as described in RFC 2617, in its 200 OK response to REGISTERs when using SIP digest. The user enables this functionality using a **sip-registrar** option.

By default, the Oracle USM supports registration with SIP digest authentication without using the authentication-info header. This is not compliant with TS 24.229. Enabling the **add-auth-info** option causes the Oracle USM to calculate and insert the required authentication-info header fields in the 200 OK.

The Oracle USM also presents this authentication header during third party registrations. The system includes the entire 200OK message in the third party registration request.

This authentication state is not shared across high availability nodes. The user can expect the Oracle USM to request re-authentication by registering UEs after failover to a backup Oracle USM.

Authentication-Info header field parameters sent by the Oracle USM include:

- **qop**—Matches the **qop** sent by the UE
- **rspauth**—A response-digest calculated as described in RFC 2617
- **cnonce**—Matches the **cnonce** sent by the UE
- **nonce-count**—Matches the **nonce-count** sent by the UE

The **nextnonce** authentication-info header field parameter, which can request a new nonce for subsequent authentication responses from the UE, is not implemented on the Oracle USM.

The ACLI syntax for enabling the **add-auth-info** option follows.

```
ORACLE(sip-registrar)#+options=add-auth-info enabled
```

The Oracle USM provides NOTICE level log entries in **log.sipd** to indicate this option's status.

Handling Barred PUIDs

The Oracle USM supports PUID barring functionality per 3GPP specification TS 24.229. As such, the system does not service any request method other than REGISTERs for SIP or Tel-

URI PUIDs designated as barred by the HSS. The Oracle USM also complies with the requirement that it allow Push Profile Requests (PPRs) to change a PUID from barred to non-barred (and vice versa) and issues a NOTIFY of the event to subscribers. No configuration is required.

A common use case for barring information is a cell phone registering with a temporary PUID (that is barred), along with a set of non-barred PUIDs in the P-Associated User (PAU) header. After registration, the cell phone should use only the non-barred PUIDs for all ensuing methods and its contacts.

An HSS should be configured with barring information for all PUIDs. During registration procedures, the HSS provides this information to the S-CSCF. PUID information in the User Data AVP of the Diameter SAA includes a tag indicating whether the PUID is barred. The Oracle USM retains this information in the registration cache. To complete the registration, the Oracle USM replies to the UE with a list of all non-barred PUIDs in the 200OK. For all the further procedures, the UE should use a PUID from the non-barred P-Associated-URI list. If the HSS does not identify a PUID's barring status, the Oracle USM assumes it is not barred.

Typical Oracle USM behaviors related to barring include:

- Responds to ensuing requests from barred PUIDs with (403) Forbidden.
- Responds to requests that have no PSU, but include barred PUIDs in their PAI header list with 403 (Forbidden).
- Responds to requests to or from wildcarded PUIDs that match barred PUIDs with 403 (Forbidden).
- Responds to registration attempts that have all barred implicit identities with 403 (Forbidden).
- Responds to requests for termination services wherein the served user (PSU/RURI) is barred with (404) Not Found.
- Recognizes barring status during third party registration procedures and does not attempt to register a barred PUID to an AS.
- Handles related subscription scenarios as follows:
  - When receiving a subscription from a barred subscriber, responds with 403 (Forbidden).
  - When receiving a subscription for a barred user, allows the SUBSCRIBE to proceed.
  - Does not include a barred identity in any NOTIFY.
    - When receiving a subscription for a user that has barred identities in its implicit set, issues NOTIFYs that only include non-barred identities.
    - Includes only non-barred PUIDs in NOTIFY messages generated by network-initiated re-registration and authorization requests.

> **Note:**
>
> The Oracle USM does not support any PUID barring within the context of GRUU.

The user can verify PUID barring status using the **show reg sipd by-user <user> detailed** command. Example output is shown below.

```
ORACLE# show reg sipd by-user user detailed
Registration Cache (Detailed View)    Thu Jul 09 2015  15:16:08
User: sip:user_1@acme-ims.com
  Registered at:  2015-07-09-15:16:04    Surrogate User: false
  Emergency Registration? No
  ContactsPerAor Rejects 0
  ContactsPerAor OverWrites 0

  Contact Information:
    Contact:
      Name: sip:user_1@acme-ims.com
      Valid: true

...

    Associated URI(s):
      URI: sip:user_1@acme-ims.com
      Status: Barred

...
```

> **Note:**
>
> The Oracle USM replicates barred status for PUIDs to standby systems.

Releasing Unregistered Users

When a call arrives at an Oracle USM either to or from a user that is not registered at that Oracle USM, it performs a location query with the HSS to determine if the unknown UE is registered at another S-CSCF. If there is no registration, the Oracle USM takes ownership of the UE. The system stores information about these UEs in its registration cache, labelled "NEVER REGISTERED". Barring any further, related action within the infrastructure, the UE would remain homed to the Oracle USM. Upon expiry of this feature's timer, the Oracle USM sends an SAR to the HSS, providing an assignment type of ADMINISTRATIVE_DEREGISTRATION for the UE. This allows the UE to be a user at a different S-CSCF the next time it is a call sender or receiver. A common use case for this scenario is a roaming UE.

When the Oracle USM issues the SAR, it also marks the UE as 'dirty' (in the process of being de-assigned) to accommodate the following operational scenarios:

- The UE attempts to register—The Oracle USM rejects the register, replying with a 504 error message.
- The UE has existing calls—The Oracle USM continues to support the call, based on a stored copy of the service profile.
- A new call arrives—The Oracle USM rejects the call. The Oracle USM replies with a '480, Temporarily Unavailable' error message if the UE is the callee; the Oracle USM responds with a 504 if the UE is the caller.

The user can configure the **unreg-cache-expiry** parameter in seconds on a per-registrar basis. This syntax is shown below.

```
ORACLE(sip-registrar)# unreg-cache-expiry 120
```

The parameter accepts values in the range of 0 to 604800, with 0 specifying that the Oracle USM does not cache unregistered users. A setting of 0 means the Oracle USM takes ownership, downloads service profiles, and then releases the user after the call without caching.

Handling Public Service Identities (PSIs)

Public Service Identities (PSI) appear as unregistered users in the Oracle USM. PSIs appear as either Distinct PSIs or Wildcarded PSIs. Similar to unregistered users, the Oracle USM takes ownership of the PSI if it is unassigned and a call is made to or from it. By default, PSIs are not released. However, the user can configure the **psi-cache-expiry** option in seconds on a per-registrar basis to cause the Oracle USM to release PSIs. This syntax is shown below.

```
ORACLE(sip-registrar)# options psi-cache-expiry=120
```

Configurable Response to Timed-Out OPTIONS Messages

The Oracle USM allows the user to configure a function by which they can cause the system to send a 408 as a response to an OPTIONS message sent to an un-responsive, registered called party. In addition, this function allows the user to specify when to send that 408.

By default, the Oracle USM does not send messages to an originating node when OPTIONS transactions time out. This complies with RFC 4321.

When registered users do not respond to OPTIONS requests, the network never informs the calling party of the called party's status. Instead, the calling party waits for the standard 32-second retry timeout to expire. If the called party was previously reachable, the calling party treats it as reachable for the entire 32-second window.

The Oracle USM includes a configuration option that:

- Starts a timer when the system forwards an applicable OPTIONS message and,
- Upon expiry of that timer, causes the system to send a 408 message to the calling party.

This option allows the network administrator to provide the calling party with this 408 response, and specify a shorter interval between request and response.

This feature works for:

- A called party that is registered via its P-CSCF, but not currently reachable.
- A called party that is reachable via an IBCF or BGCF.

This function has no impact on requests that result in a response, such as SIP 480, for un-registered subscribers.

For registered users with multiple contacts, the Oracle USM uses a response from any contact as a trigger to stop the timer and not send a 408. The Oracle USM cancels all remaining OPTIONS transactions when it receives a response from a contact. In addition, if the system used parallel forking to reach multiple contacts, it waits for the timer expiry before it sends the 200OK to the caller.

The option is available via S-CSCF processing and, as such, is available on both the Oracle USM and Oracle CSM products. There is, however, one operational difference between the Oracle USM and Oracle CSM. If the called party finally responds after this timer expires and the S-CSCF logic has sent the 408, the Oracle USM drops the response, whereas the Oracle CSM forwards it to the originating node.

The user sets the option globally in **sip-config** or on a **sip-interface**, with the **sip-interface** taking precedence. Values range from 1 to 32 seconds. Invalid ranges cause the system to use the maximum value of 32. The example below sets a sip-interface's timer to 4 seconds.

```
ORACLE(session-router)#sip-interface
ORACLE(sip-interface)#options +options-408-timeout=4
```

Option syntax on the **sip-config** and **sip-interface** configuration elements is the same.

The user must consider the infrastructure carefully. Setting the value too low can cause an inordinate number of invalid 408 responses.

Limiting REGISTER CDR Generation

The Oracle USM allows the user to generate RADIUS CDRs for REGISTER events via configuration. Large networks, however, can generate an inordinate volume of CDRs. So the Oracle USM also allows the user to reduce REGISTER CDR generation by filtering out some of the messages it sends.

When the user enables accounting with the generate-events parameter, the Oracle USM can generate CDRs for the following register and/or local register events:

- Initial REGISTER
- REGISTER refresh
- REGISTER update
- de-REGISTER

Depending on the event, the system generates per-contact start, interim and/or stop CDRs. With no other configuration, the system generates the appropriate CDRs for all of these events.

The user can prevent the system from issuing some CDR via an **account-config** option that filters, as described below, and sets a timer that restarts the CDR suppression window. Use the syntax below to set this **register-cdr-interval** option with an expiry timer value of 43200 in minutes (30 days), and limit the number of generated CDRs as described below.

```
(account-config)#options +register-cdr-interval=43200
```

When configured with this option, the Oracle USM limits the generation of CDRs for each user as follows:

1. Send a START CDR for first Register message (for first contact).
2. Don't send CDRs until the user specified time period expires. After it expires, when a Registration message causes a 'START' or 'INTERIM' CDR event to occur, send it. Then, re-set the time value. Applicable 'START' CDR events include:
    - Add new contact
    - Replace contact
    - Overwrite contact

The applicable 'INTERIM' CDR event is a Refresh Contact.

The **generate-event** parameter must also be set to **register**.

Limiting AOR Contacts

The Oracle USM allows you to limit the number of contacts that apply to AORs. It also provides a configurable behavior allowing the system to either reject a new contact or overwrite an existing contact with the new one. The user specifies the maximum number of contacts and the operation mode on a per-registrar basis. Alternatively, the user can disable the feature. This feature is applicable to Cx and local database deployments.

The value for **max-contacts-per-aor** ranges from 0-256. A value of 0 disables the function. When **max-contacts-per-aor** is greater than zero, the Oracle USM tracks the number of contacts registered per AOR. Settings for **max-contacts-per-aor-mode** include REJECT and OVERWRITE.

If you change the configured maximum while the system is operational, your setting only applies to new registrations. If there are more contacts than your newly configured maximum, the system removes older contacts. This ensures that the contacts are always within the configured maximum.

Both **max-contacts-per-aor** and **max-contacts-per-aor-mode** are RTC supported.

Maximum Contacts REJECT Mode

If the Oracle USM receives a registration request that exceeds the maximum that you configured, it responds with a local response, a 403 Forbidden by default, and does not register the additional contact. The system only rejects registration requests that exceed the maximum. Existing contacts persist normally.

Maximum Contacts OVERWRITE Mode

If the number of contacts in the initial registration exceeds the maximum, the Oracle USM selects only the highest priority contact based on q-values. If there are no q values, the Oracle USM adds contacts in the order they appear in the REGISTER message until it reaches the maximum. The system then identifies the oldest contacts for overwriting using the last registered time stamp.

In all cases, the Oracle USM follows this procedure to remove old contacts:

1. If reg-id/instance-id is present in the contact, the system simply updates the contact.
2. The system sends NOTIFY messages to the subscriber for whom the contact has been removed with a status of "terminated" and "de-activated" as the reason.
3. The system removes the contact from the registration cache.

HSS Server Assignment

As the Oracle USM registers UAs, it requests to assign itself as the S-CSCF for the registering AoR. The Oracle USM's S-CSCF identity is configured in the home-server-route parameter in sip-registrar configuration element. This is a entered as a SIP URI (containing FQDN or IP address) and is used to identify and route messages to this Oracle USM on behalf of the registered user.

Server Assignment Messages

The Oracle USM sends a Server Assignment Request (SAR) to the HSS requesting to confirm the SIP or SIPS URI of the SIP server that is currently serving the user. The SAR message also serves the purpose of requesting that the Diameter server send the user profile to the SIP server. The SAR's AVPs are populated as follows:

- Public-User-Identity—the SIP AOR of the endpoint being registered (same as UAR)
- Private-User-Identity—the username from the SIP authorization header, if it is present. If not, this value is the public User ID. (Same as UAR)
- Server-Name—the home server route parameter in the sip-registrar configuration element. It is the FQDN or IP address used to identify and route to this Oracle USM sent as a URI.
- Server-Assignment-Type—the value of this attribute depends upon the registration state:
  - REGISTRATION (1)—for all new and refreshing registrations.
  - Set to TIMEOUT_DEREGISTRATION (4)—when the contact is unregistered due to expiration. This occurs if the force-unregistration option is configured in the sip config.
  - USER_DEREGISTRATION (5)—when the contact is unregistered by the user (contact parameter expires=0).
- User-Data-Already-Available—always set to USER_DATA_ALREADY_AVAILABLE (1)

Server-Assignment-Response

The Oracle USM expects a DIAMETER_SUCCESS code in the SAA to indicate that the assignment was successful. Then a 200 OK response is returned to the registering user. Any other Diameter result code is an error and results in an error response for the original REGISTER request (by default 503) and the contacts to be invalidated in the registration cache.

Register Refresh

When a UA sends a register refresh, the Oracle USM first confirms that the authentication exists for that UE's registration cache entry, and then is valid for the REGISTER refresh. (If a valid hash does not exist for that AoR, then the Oracle USM sends an MAR to the HSS to retrieve authentication data once again).

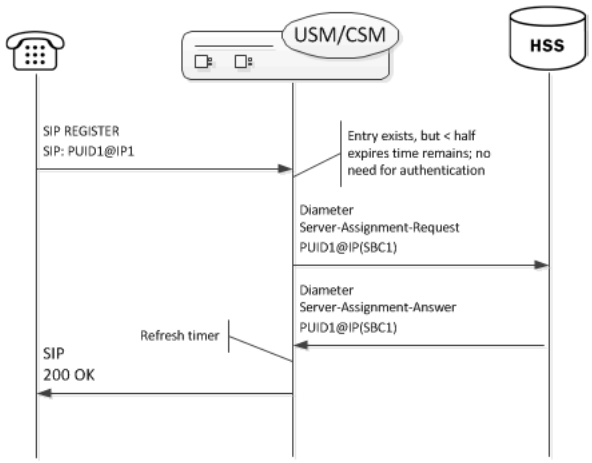
Next, the Oracle USM determines if the it can perform a local REGSITER refresh or if the HSS needs to be updated. If any of the following 3 conditions exists for the re-registering UA, the Oracle USM updates the HSS:

- The location update interval timer has expired—This value, configured in the sip registrar configuration element ensures that HSS database always has the correct Oracle USM address by periodically sending SARs for each registered contact.
- The message's call-id changes while the f**orward-reg-callid-change** option in the sip config configuration element is set. This covers the case where the UA changes the Oracle USMs through which it attaches to the network.
- The REGISTER message's Cseq has skipped a number. This covers the case in which a user registered with Oracle USM1, moves to Oracle USM2 , and then returns to Oracle USM1.

If the Oracle USM updates the HSS database because of matching one of the above conditions, the access side expiration timer per contact is reset to the REGISTER message's Expires: header value, and returned in the 200 OK. This happens even in the case when the reREGISTER was received in the first half of the previous Expires period. In addition, the core-side location update interval timer are refreshed on both active and standby.

When the above three conditions are not met, the registration expiration proceeds normally.

If the timer has not exceeded half of its lifetime, a 200 OK is returned to the UA. If the timer has exceeded half of its lifetime, the Oracle USM just refreshes the access-side expiration timer; the registration cache expiration timer for that AoR begins its count again.



Core-side SAR Lifetime

The Oracle USM maintains a timer for user registrations per SAR on the core side as specified above. The core-side SAR lifetime timer is configured in the location update interval parameter in the sip registrar configuration element. This timer ensures that the HSS always has the correct Oracle USM address, by sending SAR messages periodically.

Entry Unregistration

Because AoRs and not contacts are referenced by the HSS, an AoR is valid and should not be removed from HSS until all associated contacts have been removed or expired. If all the contacts are removed for an AoR by receiving REGISTER messages with Expires:0 header, then the SAR sent to the HSS includes Server-Assignment-Type of USER_DEREGISTRATION (5).

When the `force-unregister` option in the `sip config` is enabled, then the HSS is explicitly updated when all of the contacts for an AoR have expired. This event prompts the Oracle USM to send a SAR to the HSS using the Server-Assignment-Type of TIMEOUT_DEREGISTRATION (4).

The HSS can send a Registration-Termination-Request to request removing a registration, which corresponds to entries in the Oracle USM's registration cache. When an RTR is received, the following AVPs are expected:

- Private-User-Identity—Username of the user, which is being de-registered.
- Associated-Identities—The Private-Id's in the same subscription which need to be de-registered. (optional)

- Public-Identity—One or more public-Id's of the user being de-registered. (optional)

For the AoR specified by the Private-User-Identity AVP, all associated contacts are removed in the registration cache. The Oracle USM sends a Registration Termination Answer to the HSS to indicate success.

Diameter Message Manipulations

The Oracle USM can perform manipulations on all grouped and non-grouped AVPs. This is referred to as Diameter Manipulation Rules (DMR). A message manipulation is the ability to search for a predefined string within an AVP and then replace it with another value. This is similar to the Oracle USM's header manipulation rules functionality.

A diameter manipulation configuration element is defined by a name parameter. You can optionally add a description field to the diameter manipulation. Within each diameter manipulation you can configure multiple diam manipulation rule subelements. The manipulation rule subelements are the configuration where AVPs are identified, searched, and in which the data is replaced.

The Oracle USM supports diameter manipulation across the Cx interface, with the user configuring these manipulations to home subscriber server configurations.

> **Note:**
>
> The user can also apply diameter manipulations to external policy server configurations. These manipulations affect traffic between the Oracle USM and the applicable policy server. The range of manipulation supported over the Rx interface is the same as that over the Cx interface.

See a full explanation on diameter manipulation, including configuration instructions, in the *External Policy Servers* chapter of this document.

User Registration based on Reg-ID and Instance-ID (RFC 5626)

Sometimes a user's device reregisters from a different network than its original registration. This event should be considered a location update rather that a completely new registration for the Contact. The Oracle USM can perform this way by considering the endpoint's reg-id and instance-id parameters defined in RFC 5626.

The Oracle USM identifies new REGISTER requests received on a different access network as a location update of the existing binding between the Contact and AoR. Without this feature, the Oracle USM would create a new binding and leave the old binding untouched in the local registration cache/ENUM database. This scenario is undesirable and leads to unnecessary load on various network elements including the Oracle USM itself.

The following conditions must be matched to equate a newly registering contact as a location update:

For a received REGISTER:

- The message must not have more than 1 Contact header while 1 of those Contact headers includes a reg-id parameter. (failure to pass this condition prompts the Oracle USM to reply to the requester with a 400 Bad Request).
- The Supported: header contains **outbound** value
- The Contact header contains a **reg-id** parameter
- The Contact header contains a **+sip.instance** parameter

After these steps are affirmed, the Oracle USM determines if it is the First hop. If there is only one Via: header in the REGISTER, the Oracle USM determines it is the first hop and continues to perform Outbound Registration Binding processing.

If there is more than 1 Via: header in the REGISTER message, the Oracle USM performs additional validation by checking that a Path: header corresponding to the last Via: includes an ob URI parameter, Outbound Registration Binding may continue.

If the Oracle USM is neither the first hop nor finds an ob URI in Path headers, it replies to the UA's REGISTER with a 439 First Hop Lack Outbound Support reply.

reREGISTER Example

The user (AoR) bob@example.com registers from a device +sip.instance= <urn:uuid:0001> with a reg-id ="1", contact URI = sip:1.1.1.1:5060. A binding is be created for bob@example.com+<urn:uuid:0001>+reg-id=1 at sip:1.1.1.1.:5060.

Next, Bob@example.com sends a reREGISTER with the same instance-id but with a different reg-id = 2 and contact URI = sip:2.2.2.2:5060.

The previous binding is removed. A binding for the new contact URI and reg-id is created. bob@example.com+<urn:uuid:0001>+reg-id=2 at sip:2.2.2.2:5060

Outbound Registration Binding Processing

An outbound registration binding is created between the AoR, instance-id, reg-id, Contact URI, and other contact parameters. This binding also stores the Path: header.

Matching re-registrations update the local registration cache as expected. REGISTER messages are replied to including a Require: header containing the outbound option-tag.

If the Oracle USM receives requests for the same AOR with some registrations with reg-id + instance-id and some without them, the Oracle USM will store them both as separate Contacts for the AOR; The AoR+sip.instance+reg-id combination becomes the key to this entry.

Wildcarded PUID Support

The Oracle USM supports the use of wildcarded Public User IDs (PUIDs), typically for registering multiple endpoints on a PBX with a single PUID. A wildcard is composed of a regular expression that, when used in a PUID prefix, represents multiple UEs. The group of UEs is referred to as an implicit registration set and share a single service profile. This support is typically implemented to reduce HSS resource requirements. The regular expressions themselves are in form of Perl Compatible Extended Regular Expressions (PCRE).

Each implicit registration set is associated with an explicitly registered distinct PUID. Typically, this distinct PUID is the PBX itself. The implicit registration set is dependent on the distinct PUID, including the distinct PUID's registration status.

There is no Oracle USM configuration required.

Wildcarded PUID support is applicable to both I-CSCF and S-CSCF operation. In addition, all Oracle USMs in the applicable data paths must be in the same trust domain.

To allow the feature, the Oracle USM supports:

- Wildcarded PUID AVP in the LIR, SAR and SAA
- User Profile AVP in the SAA
- P-Profile-Key across the Mw interface, as defined in RFC 5002

Note also that the HSS must support the wildcarded-public-Identify AVP.

ACLI Instructions

The following configuration enables the Oracle USM to authorize and authenticate registering users. In addition it sets the Oracle USM to request itself as the S-CSCF for the registering users.

home subscriber server

To configure a home subscriber server (HSS):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **home-subscriber-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# home-subscriber-server
   ORACLE(home-subscriber-server)#
   ```

4. **name**—Enter the name for this home subscriber server configuration element to reference from other configuration elements.
5. **state**—Set this to **enabled** to use this configuration element.
6. **address**—Enter the IP address of this HSS. Both IPv4 and IPv6 addressing is supported.
7. **port**—Enter the port which to connect on of this HSS, the default value is 80.
8. **realm**—Enter the realm name where this HSS exists.
9. Type **done** when finished.

SIP Authentication Profile

To configure the SIP Authentication Profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-authentication-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-authentication-profile
   ORACLE(sip-authentication-profile)#
   ```

   You may now begin configuring the SIP Authentication Profile configuration element.

4. **name**—Enter the name of this SIP authentication profile that will be referenced from a SIP registrar (or a SIP interface) configuration element.
5. **methods**—Enter all the methods that should be authenticated. Enclose multiple methods in quotes and separated by commas.
6. **anonymous-methods**—Enter the methods from anonymous users that require authentication. Enclose multiple methods in quotes and separated by commas.
7. **digest-realm**—Leave this blank for Cx deployments.
8. **credential-retrieval-method**—Enter CX.
9. **credential-retrieval-config**—Enter the home-subscriber-server name used for retrieving authentication data.
10. Type **done** when finished.

SIP Interface

The full SIP interface should be configured according to your network needs. Please refer to the Oracle SBC ACLI Configuration Guide.

To configure a SIP Digest Authentication on a specific SIP Interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and choose the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(sip-interface)# select
   <realm-id>:
   1: private 192.168.101.17:5060
   2: public  172.16.101.17:5060
   selection: 1
   ```

5. **registration-caching**—Set this parameter to **enabled.**
6. **ims-access**—Set this parameter to **enabled** for access interfaces, when applicable. Core interfaces should have this feature disabled.
7. **sip-authentication-profile**—Set this to the name of an existing sip-authentication profile if you wish to authenticate per SIP interface.
8. Type **done** when finished.

SIP Registrar

To configure the Oracle USM to act as a SIP Registrar:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)#
   ```

4. **name**—Enter a name for this SIP registrar configuration element.
5. **state**—Set this to **enabled** to use this SIP registrar configuration element.
6. **domains**—Enter one or more domains that this configuration element will invoke SIP registration for. Wildcards are valid for this parameter. Multiple entries can be entered in quotes, separated by commas.
7. **subscriber-database-method**—Set this to **CX**.
8. **subscriber-database-config**—Enter the home-subscriber-server configuration element name that will handle REGISTER messages for this domain. The HSS configuration element includes the actual IP address of the server that SAR's are sent to.
9. **authentication-profile**—Enter a sip-authentication-profile configuration element's name. The sip authentication profile object referenced here will be looked up for a REGISTER message with a matching domain in the request URI. You may also leave this blank for the receiving SIP Interface to handle which messages require authentication if so configured.
10. **home-server-route**—Enter the identification for this Oracle USM that will be sent as the Server-Name in MAR and SAR messages to the HSS. This value should be entered as a SIP URI.
11. **location-update-interval**—Keep or change from the default of 1400 minutes (1 day). This value is used as the timer lifetime for core-side HSS updates.
12. Type **done** when finished.

Maximum Number of Contacts

To configure a sip-registrar with a maximum of 10 contacts per AOR and a mode of overwrite:

1. From superuser mode, use the following command sequence to access sip-registrar element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)# select
   ```

   Select the registrar you want to configure.

2. Specify the number of contacts.

```
ORACLE(sip-registrar)# max-contacts-per-aor 10
ORACLE
```

3. Specify the contact mode to overwrite.

```
ORACLE(sip-registrar)# max-contacts-per-aor-mode overwrite
ORACLE
```

4. Type **done** and **exit** to complete configuration of this **sip-registrar** configuration element.

Response to Exceeding Maximum Contacts

To configure local response for the Oracle USM to issue when max-contacts-per-aor is exceeded:

1. From superuser mode, use the following command sequence to access local-response and add an entry.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# local-response-map
```

2. Access the entries configuration.

```
ORACLE(local-response-map)# entries
```

3. Specify the local error you need to configure.

```
ORACLE(local-response-map-entry)# local-error contacts-per-aor-exceed
```

4. Specify the sip-reason for this error.

```
ORACLE(local-response-map-entry)# sip-reason forbidden
```

5. Specify the error code for this error.

```
ORACLE(local-response-map-entry)# sip-status 403
ORACLE(local-response-map-entry)# done
local-response-map-entry
        local-error                     contacts-per-aor-exceed
        sip-status                      403
        q850-cause                      0
        sip-reason                      forbidden
        q850-reason
        method
        register-response-expires
ORACLE(local-response-map-entry)# exit
```

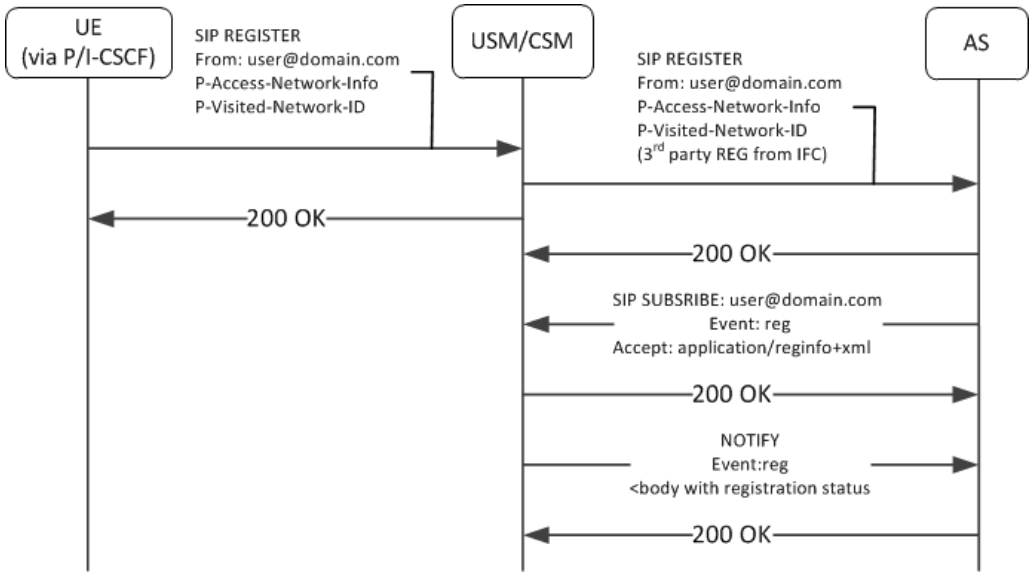
SIP Registration Event Package Support

The Oracle USM supports UA subscriptions to the registration event package, as defined in RFC3680. As such, it maintains contact with entities, often application servers, that need to know about UA registration events and provides those application servers with notifications when registration events occur.

Common usage for this functionality includes:

- Forcing re-authentication
- The provision of welcome notices to users who need information or instructions customized to their location

An operational example, shown below, begins with the Oracle USM performing 3rd party registration on behalf of a UA to an AS, based on the iFC request from the UA. The AS, being an appropriately authorized UA itself, subscribes to NOTIFY messages on reg events for the initial UA. The Oracle USM sends a 200OK to the AS, and then proceeds to forward NOTIFY messages about that UE's registration events to the AS.



This feature is relevant when the Oracle USM is performing S-CSCF functions. You enable this feature on the Oracle USM per registrar, by simply creating a profile and applying it to the applicable registrar.

SUBSCRIBE Processing

When the Oracle USM has the reg-event notification function enabled for a registrar, it includes the allow-events header in its 200OK replies to successful REGISTERS. This lets UEs know that they can subscribe to registration event packages.

When the Oracle USM receives reg-event subscription requests, it follows the sequence below to process SUBSCRIBE requests for reg events:

1. Determines validity of the request.

   Subscriptions cannot include more than one package name. If there is more than one package name in the request, the Oracle USM replies with a 400 Bad Request message.

2. Determines if it can be a notifier, as follows:

   - The SUBSCRIBE must include EVENT=reg.
   - The requesting UA must be in the same domain as the registrar.

   If both of the above are true, the Oracle USM proceeds with the request.

3. Authorizes the request. The Oracle USM only authorizes requests from UEs that come from the same realm and layer 2 connection on which it received the initial REGISTER. Furthermore, the Oracle USM only authorizes the following UEs:

   - Public user identities from UEs that are subscribing to their own registration events.

- Public user identities that this user owns. Examples include implicitly registered public user identities.
- Entities that were included in the PATH header of the target UE's registration.
- All ASs that are listed in the UE's iFC and that are part of the trust domain.

If all of the above are true, the Oracle USM proceeds with the request. If not, it sends 403 Forbidden to the requester.

4. Determines how it is functionally related to the UA. The Oracle USM only processes subscriptions for users in its registration cache, replying with a 403 Forbidden if not. For cached users, the Oracle USM forwards the request to the registrar if it is the P-CSCF. If it is the S-CSCF, it sends a 200 OK and begins to act as notifier.
5. Identifies the subscription duration, as follows, and sends the 200 OK to the UE:

   If there is no Expires header in the UE's 200OK message, the Oracle USM applies it's own configured minimum or the default (600000 seconds), whichever is greater.

   If the SUBSCRIBE includes an Expires header, the Oracle USM honors the request unless it is less than the configured minimum.

   If the SUBSCRIBE's Expires header is less than the minimum subscription time configured in the registration event profile, the Oracle USM denies the subscription, sending a 423 Too Brief message.

   When the Oracle USM encounters an Expires header set to 0, it terminates the subscription. This is referred to as unsubscribing.

SUBSCRIBE REFRESH Requests

Subscriptions must be refreshed to keep them from expiring. ASs accomplish this by sending SUBSCRIBE REFRESH messages to the Oracle USM. Messages must be received from authorized subscribers and on the same realm and connection as the original SUBSCRIBE or the Oracle USM rejects the refresh request.

Reg Event NOTIFY Messages

When configured, the Oracle USM issues NOTIFY messages to subscribed ASs when significant registration events occur. NOTIFY messages sent by the Oracle USM comply fully with RFC3680. Events that trigger NOTIFY messages include:

- Registered
- Registration refreshed
- Registration expired
- Registration deactivated
- UE unregistered

The Oracle USM does not send NOTIFY messages for the following events:

- Registration created
- Registration shortened
- Registration probation
- Registration rejected

Additional detail about NOTIFY messages that is specific to the Oracle USM includes:

- The Oracle USM always sends full information on all contacts, and indicates such within the reginfo element. The Oracle USM does not utilize the partial state described within RFC 3680.
- Wildcarded PUIDs are included, enclosed in the <wildcardedIdentity> tag within the <registration> element.
- The Oracle USM does not include the following optional attributes within the contact element:
  - expires
  - retry-after
  - duration registered
  - display-name
- The Oracle USM uses the optional unknown-param element within the contact element to convey UA capabilities and distribute reg-id, sip.instance and header filed attributes.

An example of the XML body of a NOTIFY message below documents the registration status for the AOR joe@example.com.

```
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo"   xmlns:xsi=http://www.w3.org/
2001/XMLSchema-instance version="0" state="full">
  <registration aor="sip:joe@example.com" id="as9" state="active">
    <contact id="6" state="active" event="registered">
      <uri>sip:joe@pc887.example.com</uri>
    </contact>
      <contact id="7" state="terminated" event="expired">
      <uri>sip:joe@university.edu</uri>
    </contact>
  </registration>
</reginfo>
```

Use the show registration and show sipd subscription commands to display all information about each subscription.

Reducing NOTIFY Traffic

RFC 3265 stipulates that the Subscription server sends NOTIFY messages to all subscribers when a UA sends a registration refresh. This can generate excessive NOTIFY traffic. You, however, can mitigate this by configuring the Oracle USM to limit notification traffic. By specifying the number of seconds between NOTIFY messages, you prevent the Oracle USM from sending notifications upon events that do not generate a change in the registration database.

Database changes that trigger notifications when this option is configured include:

- The Cseq number of the REGISTER message increases by more than 1
- The call-ID changes
- A contact parameter changes
- The number of contacts changes

Upon expiry of this timer, the Oracle USM sends out a NOTIFY for every registration event subscription. Note also that the Oracle USM does not send the cseq attribute in the CONTACT element when this interval is configured.

Configuring Registration Event Package

This section shows you how to create reg-event profiles and apply those profiles to sip-registrars. These profiles enable the monitoring of UA registration events and the delivery of state change notifications to each UA that subscribes to the package. The procedure includes:

- Create one or more registration-event profiles
- Apply each profile to the applicable sip-registrar
- Optionally specify the registration event notification interval timer

Registration Event Profile Configuration

To configure a registration event profile:

1. From superuser mode, use the following command sequence to access regevent-notification-profile command.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# regevent-notification-profile
ORACLE(registration-event-profile)#
```

2. To define the profile, simply name it and specify a timeout in seconds.

```
ORACLE(registration-event-profile)# name reg-event-profile1
ORACLE(registration-event-profile)# min-subscription-duration 2500
ORACLE(registration-event-profile)# done
ORACLE(registration-event-profile)# exit
```

3. Navigate to the registrar for which you want registration event package support.

```
ORACLE(session-router)# sip-registrar
ORACLE(sip-registrar)# regevent-notification-profile reg-event-profile1
ORACLE(sip-registrar)# done
ORACLE(sip-registrar)# exit
```

Optional NOTIFY Refresh Frequency

To specify optional NOTIFY refresh frequency:

1. From superuser mode, use the following command sequence to access registration-event-profile command within session router.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# regevent-notification-profile
ORACLE(registration-event-profile)#
```

2. To enable NOTIFY, set the send-notify-for-reg-refresh option to the time, in seconds,

```
ORACLE(registration-event-profile)# options notify-refresh-interval=1800
ORACLE(registration-event-profile)# done
ORACLE(registration-event-profile)# exit
```

Prepend the option with the + sign if you have multiple options configured that you want to retain.

```
ORACLE(registration-event-profile)# options +notify-refresh-interval=1800
```

Running the command without the + character causes the system to remove any previously configured options.

Message Routing

The Oracle USM provides two major types of routing that use the routing precedence parameter in the sip registrar. Routing precedence can be set to either **registrar** (HSS) or **local policy**. Routing precedence is set to registrar by default. There are additional controls that the user may configure to refine message routing.

Registrar routing uses the configured subscriber database and registration cache to route the call. Local policy routing lets you configure routing decisions within the Oracle USM's local policy routing functionality.

Within the context of local policy routing, the Oracle USM chooses the next hop through the network for each SIP session based on information received from routing policies and constraints. Routing policies can be as simple as routing all traffic to a proxy or routing all traffic from one network to another. Routing policies can also be more detailed, using constraints to manage the volume and rate of traffic that can be routed to a specific network. For example, you can manage volume and rate of traffic to enable theOracle USM to load balance and route around softswitch failures.

When a message arrives at the Oracle USM, it determines whether it is coming from a session agent. If so, the Oracle USM checks whether that session agent is authorized to make the call. Local policy is then checked to determine where to send the message.

Depending on whether the Oracle USM is performing originating or terminating services for the call, described in the chapter on operations within the IMS core, it performs those services prior to routing to the endpoint.

If the Oracle USM is unable to proceed with routing a request, it replies to the UA that sent the request with a 4xx response.

This chapter provides an overview of registrar routing for perspective, but focuses on local policy routing. Local policy routing is configuration intensive, allowing precise route specification. As a result, configuring local policy routing is a complex process requiring that the user understand the purpose and interaction of multiple configuration elements. This chapter also provides descriptions and configuration instruction on additional routing controls, such as the use of multistage and UA capability routing.

Registrar Routing

When the routing precedence parameter is set to **registrar**, the Oracle USM is using the HSS as a resource within the context of its routing decisions.

When an INVITE arrives, the Oracle USM checks its own registration cache for a pre-existing matching contact in the INVITE. If it finds a match, it forwards the request to that location. If it does not find a match, it issues an Location Information Request (LIR) to the HSS. If the HSS's response, called an LIA, provides an assigned S-CSCF for that UA, the Oracle USM proceeds as described below in the section LIR/LIA Transaction.

Note that you can configure the Oracle USM to fallback to a local policy lookup if the lookup via the registrar fails. Configure this by adding the **fallback-to-localpolicy** option to the sip-registrar configuration element.

For situations where the database routing decision needs to be done in lieu of the default, you can set routing precedence to local-policy. Note that you can configure a routing entry that

points to an HSS by setting a policy attribute with a next-hop of cx:<home-subscriber-server-name> within the local-policy.

LIR/LIA Transaction

An LIR includes the Public-User-Identity AVP, which contain a UA's actual PUID. The HSS responds with the assigned S-CSCF server (often a Oracle USM) for this PUID. The answer is the form of a Location Info Answer (LIA). The LIA includes the assigned S-CSCF in the Server Name AVP.

If the S-CSCF returned in the LIR is this Oracle USM, then the Oracle USM performs unregistered termination services for this UA. (This situation indicates that the UA is currently unregistered.) Such services could include directing the call to voice mail. If the HSS returns an S-CSCF in the LIA that is not this Oracle USM, it forwards the request to that S-CSCF.

Default Egress Realm

The sip registrar configuration element should be configured with a default egress realm id. This is the name of the realm config that defines the IMS control plane, through which all Oracle USMs, HSSs, and other network elements communicate and exchange SIP messaging. It is advisable to configure this parameter to ensure well defined reachability among Oracle USMs.

RX Interface Features

The Oracle USM can run the Rx interface over a Diameter connection and act as a P-CSCF communicating with a PCRF. The Rx interface supports quality of service and policy management within applicable network infrastructures. See the Oracle SBC ACLI Configuration Guide for full descriptions of this functionality.

ACLI Instructions

Configuring the SIP Registrar's Routing Precedence

To configure a SIP registrar configuration element for message routing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)#
   ```

4. Type **select** and choose the number of the pre-configured sip interface you want to configure.
5. **routing-precedence**— Set this to either **registrar** or **local-policy** depending on your deployment.
6. **egress-realm-id**—Enter the default egress realm for Oracle USM messaging.

7. Type **done** when finished.

Home Subscriber Server

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **home-subscriber-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# home-subscriber-server
   ORACLE(home-subscriber-server)#
   ```

4. Begin configuring your HSS, or type **select** and choose the number of the pre-configured HSS you want to configure.
5. Type **done** when finished.

Tel-URI Resolution

The Oracle USM can initiate number resolution procedures for requests that have tel-URI or SIP-URI (with user=phone) numbers in the R-URI. It does this by querying number resolutions services, including the local routing table(s) or ENUM server(s) to resolve the R-URI to a SIP URI. In addition, the original R-URI may not include a full E.164 number. As such, you can also configure the Oracle USM to perform a number normalization procedure and ensure it presents a full E.164 number for resolution. Upon successful resolution, the Oracle USM proceeds with ensuing signaling procedures.

To configure the Oracle USM to perform these lookups, you create applicable **local-routing-config** or **enum-config** elements and set an option within the **sip-registrar** that specifies a primary and, optionally, a secondary **local-routing-config** or **enum-config** that the **sip-registrar** uses for LRT or ENUM lookups. If there is no ENUM configuration on the **sip-registrar**, the Oracle USM forwards applicable requests to a border gateway function via local policy.

Refer to the *Oracle Communications Session Border Controller ACLI Configuration Guide* , Session Routing and Load Balancing chapter for complete information on how to configure a **local-routing-config** and/or an **enum-config**.

Number Lookup Triggers

Use cases that are applicable to number lookups and the associated Oracle USM procedures include:

- Request from the access side:
  1. The Oracle USM performs originating services.
  2. If the R-URI is a tel-URI or SIP-URI (with user=phone), it requests e.164 resolution from the ENUM server(s), regardless of its presence in the registration cache.
- Request from core side including request for originating services:
  1. The Oracle USM performs originating services.

2. If the R-URI is a tel-URI or SIP-URI (with user=phone), it requests e.164 resolution from the ENUM server(s), regardless of its presence in the registration cache.

- Request from core side, for terminating services only:
  1. If the R-URI is a tel-URI or SIP-URI (with user=phone) and is not in the Oracle USM cache, it performs an LIR.
  2. If the LIA reply indicates the tel-URI or SIP-URI (with user=phone) is not provisioned, the Oracle USM requests e.164 resolution from the ENUM server(s).

Actions Based on Lookup Results

The Oracle USM forwards to the resultant SIP-URI under the following conditions:

- The SIP-URI is in the Oracle USM cache, in which case the Oracle USM performs terminating services.
- The SIP-URI is not in the Oracle USM cache, and the Oracle USM is configured to service the returned domain.
  In this case, the Oracle USM performs the following:
  1. The Oracle USM issues an LIR for the SIP-URI.
  2. The Oracle USM forwards the message to the correct S-CSCF.
- The SIP-URI is not in the Oracle USM cache, and the Oracle USM is not configured to service the returned domain.
  In this case, the Oracle USM performs refers to local policy to forward the message via local policy.

PSTN Breakout Routing

The Oracle USM complies with RFC 4694 for operation with request-URIs that include carrier identification code/route number/number portability database dip indicator (cic/rn/npdi) information and routes those requests according to the rn information. The routing process includes utilization of local policy configured to break the request out of the home network via gateways such as a BGCF.

The Oracle USM does not validate any rn or cic information. Instead, it simply routes the request. Note that the Oracle USM uses cic information instead of rn if both are present in the request. RFC 4694 compliant circumstances under which the Oracle USM does not use rn, cic and npdi information include:

- Invalid routing information, including rn present, but npdi missing.
- Invalid routing information, including npdi present, but rn missing.
- Request uses a `sip-URI` presented without `user=phone`.

If the request includes originating services as well as cic/rn/npdi information, the Oracle USM performs those services rather than break out. If, after completing originating services, the request still includes cic/rn/npdi information, the system performs this breakout.

Primary and Secondary ENUM Configuration

For the purpose of redundancy, the Oracle USM allows you to configure these number lookups to use a backup resource in case the lookup at the primary fails. Such scenarios include losing contact with the primary ENUM/LRT server config (query time-out) and the entry is not found at the primary (LRT or ENUM).

To apply primary and secondary number lookup resources to a sip-registrar:

1. From superuser mode, use the following command sequence to access the sip-registrar element and select the registrar you want to configure.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)# select
   ```

2. Specify the resources to use with the options command.

   Prepend the option with the + character if you have multiple options configured that you want to retain. Running the command without the + character causes the system to disable any previously configured options.

   To specify primary and secondary ENUM servers:

   ```
   ORACLE(sip-registrar)# options +e164-primary-config=enum:<enum-config name>
   ORACLE(sip-registrar)# options +e164-secondary-config=enum:<enum-config name>
   ORACLE(sip-registrar)# done
   ```

   To specify primary and secondary LRT resources:

   ```
   ORACLE(sip-registrar)# options +e164-primary-config=lrt:<lrt-config name>
   ORACLE(sip-registrar)# options +e164-secondary-config=lrt:<lrt-config name>
   ORACLE(sip-registrar)# done
   ```

   Bear in mind that an enum-config can reference multiple servers. When the Oracle USM references an enum-config, queries follow the normal enum-config sequence, checking each referenced server in order. If the lookup is not successful at the primary, the Oracle USM checks the servers in the registrar's e164-secondary-config.

   In addition, each enum-config may refer to a different top-level-domain. This allows you to configure the Oracle USM to successfully perform lookups within two domains.

HSS Initiated User Profile Changes

The Oracle USM can receive Push Profile Request (PPR) messages from an HSS and update the state of the IMS User Profile and associated subscription information it has cached locally. The SIP digest authentication information can also be updated and reassociated with an AoR in case that has changed too. The Oracle USM expects to receive the following AVPs in a PPR message.

- Private-User-Identity—the username, whose subscription/authentication data has changed.
- SIP-Auth-Data-Item—if present new authentication data is included here.
- User-Data—if present new User data is included here.
- Charging-Information—if present new charging information is included here.

The Oracle USM replies to an HSS's PPR in a PPA message with the following AVPs:

- Result-Code—indicates Diameter base protocol error. Valid errors for in a PPA are:
  - DIAMETER_SUCCESS—The request succeeded.
  - DIAMETER_ERROR_NOT_SUPPORTED_USER_DATA—The request failed. The Oracle USM informs HSS that the received user information contained information, which was not recognized or supported.

  - DIAMETER_ERROR_USER_UNKNOWN—The request failed because the Private Identity is not found in Oracle USM.
  - DIAMETER_ERROR_TOO_MUCH_DATA—The request failed. The Oracle USM informs to the HSS that it tried to push too much data into the Oracle USM.
  - DIAMETER_UNABLE_TO_COMPLY—The request failed.
- Experimental-Result—indicates diameter application (3GPP/Cx) error if present.

Licensing and Database Registration Limits

The Oracle USM limits the number of unexpired registration cache entries globally. The total number of system registrations is configured with the registration cache limit parameter in the sip config configuration element.

The Oracle USM also limits the number of registration cache entries that were obtained from a User Subscriber Database; only REGISTERs that prompted the database query are counted here. As User Subscriber Database entries are added and removed, this counter is updated accordingly. Note that it is the actual number of SD-contacts that count against the license limit. Discrete database registration license values range from 20,000 through 500,000 in increments of 20,000.

When a registering contact is rejected because it will exceed one of these limits, the Oracle USM sends a 503 message to the registering endpoint.

Refer to the Getting Started chapter for information about install license management.

Database Registration Limit Alarm

By default, a major alarm is enabled when 98% or more of the licensed number of Database Registrations are used. This alarm is cleared when the number of database registrations falls below 90%. You can configure minor and critical alarms when crossing configured thresholds and you can also reassign the major alarm. This is configured in by creating a **system-config** > **alarm-threshold** sub element with type of **database-registration**.

3GPP Compliance

P-Asserted-Id in Requests and Dialogs

When an AoR is successfully registered through the Oracle USM, the list of implicitly registered public IDs is returned from the HSS. The set of implicitly registered public IDs includes the explicitly registered Public-id and may include wild-carded public-ids. If there are no implicitly registered public-ids, then the implicit set returned by HSS will at least contain the explicitly registered public-id.

Based on local configuration and network conditions, the registering UE may or may not be trusted.

Non-trusted UE

When a non-trusted UE sends an initial request for a dialog or a request for a standalone transaction to the Oracle USM, the P-Asserted-Identity to be inserted in the outgoing request is formed as follows:

- If the request does not have a P-Preferred-Identity header field or none of the P-Preferred-Identity header fields included in the request match any of the registered public user identities, then the Oracle USM inserts the default Public-Identity in the outgoing P-Asserted-Identity header.
- If the request includes one or more P-Preferred-Identity header fields which match the registered public user identities, then the Oracle USM includes only the first P-Preferred-Identity header field.
- If the request includes one or more P-Asserted-Identity header fields which do not match the registered public user identities, then the Oracle USM inserts the default Public-Identity in the outgoing P-Asserted-Identity header.

Trusted UE

When a trusted UE sends an initial request for a dialog or a request for a standalone transaction to the Oracle USM, the P-Asserted-Identity to be inserted in the outgoing request is formed as follows:

- If the request does not have a P-Preferred-Identity header field or none of the P-Preferred-Identity header fields included in the request match any of the registered public user identities, then the Oracle USM inserts the default Public-Identity in the outgoing P-Asserted-Identity header.
- If the request includes one or more P-Preferred-Identity header fields which match the registered public user identities, then the Oracle USM inserts the first P-Preferred-Identity header field.
- If the request includes one or more P-Asserted-Identity header fields, then the Oracle USM will use the first P-Asserted-Identity header field.
  - The contents of the From header field do not form any part of this decision process.
  - P-Preferred-Identity header fields will always be removed.

The Default Public Identity is the first appearing, non-barred identity in the set of implicitly registered Public User Identities.

To enable this behavior, add the **pai-comply-to-3gpp** option in the sip config configuration element.

P-Associated-URI in 200 OK

In a 200 OK response to a UE on a successful registration, the Oracle USM includes a P-Associated-URI header. This header includes the list of registered, distinct public user identities and the associated set of implicitly registered distinct public user identities.

When there are no associated implicit public identities, only the explicitly registered Public User Identity is included.

Other Diameter Cx Configuration

Host and Realm AVP Configuration for Cx

You can configure the values sent in the origin-host, origin-realm and destination-host AVPs when the Oracle USM communicates with a server over the Cx interface. Configure

destination-host when you want to precisely specify the HSS with which these Cx exchanges take place.

The applicable configuration parameters are located in the home-subscriber-server configuration element. The parameters used to configured the AVPs are origin-realm, origin-host-identifier and destination-host-identifier. The AVPs are constructed as follows:

```
Origin Host AVP = <origin-host-identifier>.<origin-realm>
Origin Realm AVP = <origin-realm>
Destination Host AVP = <destination-host-identifier>.<destination-realm>
```

If the **origin-realm** is not configured, then the realm parameter in the home-subscriber-server configuration element will be used as the default. If **origin-host-identifier** is not configured, then the name parameter in the home-subscriber-server configuration element will be used as the default.

If these parameters are not configured, then the AVPs are constructed as follows:

```
Origin Host = <HSS Config name>.<HSS Config realm>.com
Origin Realm AVP = <HSS Config realm>
Destination Host = <HSS Config name>.<HSS Config realm>.com
```

ACLI Instructions

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **home-subscriber-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# home-subscriber-server
   ORACLE(home-subscriber-server)#
   ```

4. **origin-realm**—Set this to a string for use in constructing unique Origin Host and Origin Realm AVPs.
5. **origin-host-identifier**—Set this to a string for use in constructing a unique Origin Host AVP.
6. **destination-host-identifier**—Set this to a string for use in constructing a unique Destination Host AVP.
7. Save your work.

Initial Filter Criteria (IFC)

The Oracle USM, acting as a S-CSCF, downloads a set of rules known as Initial Filter Criteria (IFC) from the HSS/AS. IFCs are downloaded over the Cx interface.

iFCs are a way for an S-CSCF to evaluate which ASs should be involved in the call flow for a given user agent (UA). iFCs are functionally defined by Boolean statements, whose component parts are expressed in XML; they reference the destination AS(s) where a desired service is provided.

IFC Evaluation

IFCs are evaluated as described in 3GPP TS 29.228. The Oracle USM supports all tags found in the 3GPP initial filter criteria specifications. An IFC is evaluated until its end, after which the call flow continues as expected.

SIP Registration

When the Oracle USM receives an authenticated REGISTER request from a served UA, it sends an SAR request to the HSS to obtain an SAA which includes iFCs associated with the UE's subscription. Within the context of registration, the Oracle USM also manages third party registration procedures in conjunction with iFC exchanges or manually via the ACLI. These procedures are described in the Third Party Registration chapter.

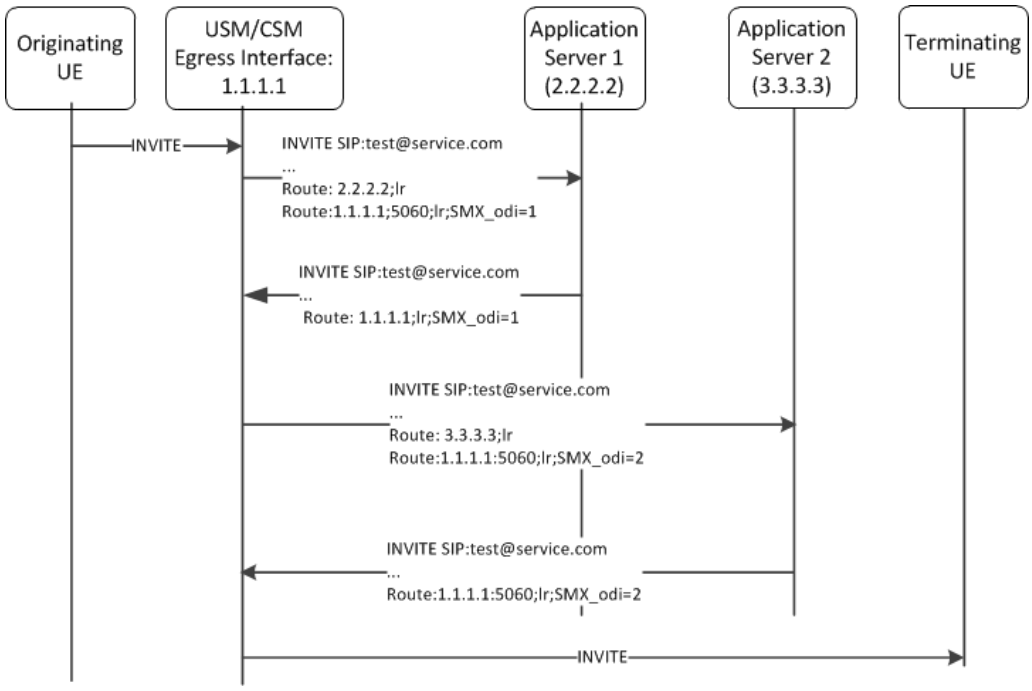
SIP Call

The Oracle USM evaluates all IFC logic to determine that messages with matching characteristics are sent to the proper AS specified in the iFC evaluation using the IP Multimedia Service Control (ISC) interface. In this INVITE, the Oracle USM adds two Route headers. The first (top) route header contains the target AS's URI. The second Route parameter is built using the IP address of the egress SIP interface and contains the ODI as a parameter. For example:

```
INVITE SIP:test@service.com
...
Route:2.2.2.2;lr
Route:1.1.1.1:5060;lr;smx_odi=1
```

If the AS routes the call back to the Oracle USM, it is expected to include the ODI parameter that it received from the Oracle USM, unchanged. The presence of the ODI parameter indicates that IFC evaluation needs to continue from where it left off for this call. If this continuation of IFC evaluation results in another AS URI, the Oracle USM initiates a request towards that AS this time with a new ODI. In this way, the ODI is a state-signifier of Service Point Triggers.

The process continues until IFC evaluation is completed. Below is an example of an IFC evaluation completing after two iterations.



The iFC continues to be evaluated completely which may result in the INVITE being forwarded to additional ASs. At the conclusion of evaluating the iFC, the Oracle USM checks if the target of the initial request is registered to itself, or not. If the UA is not registered locally the Oracle USM forwards the request by regular means into the network. If the target UA is registered locally, the Oracle USM proceeds to obtain iFCs for the target and begin iFC evaluation for the terminating side of the call.

Evaluating Session Case in the P-Served-User Header

The P-served-user header field conveys the identity of the served user, the session case that applies to the particular communication session, and application invocation, as defined in RFC 5502 and TS 24.229. The Session Case (sescase) and Registration State (regstate) parameters are either populated by the UA originating the message or by the Oracle USM after it determines if a request is originating or terminating, and registered or unregistered

The P-served-user header is created and added to an outgoing request if the next hop is trusted. A trusted next hop is an entity defined by a session agent whose trust-me parameter is enabled. Likewise, the P-served-user header is stripped if the request is forwarded to a next hop that is not known to be trusted.

When the Oracle USM creates a P-served-User header, the user value in the originating case is the user value found in the P-Asserted-Identity header field. In the terminating case, the user value is taken from the Request URI.

Supported Sessioncase and Registration State

The following cases are supported for IFC evaluation. Conditions for classifying the calls as such are listed below.

Originating request - Registered User

When the Oracle USM receives an Initial request, it is validated as an originating request from or on behalf of a registered user when the following conditions are met:

- When the **ignore-psu-sesscase** option is not set:
  - The request is a dialog creating request or a standalone request.
  - There is no "odi" parameter in the top route of the request.
  - The regstate and sescase parameters of the P-served-user indicate for this to be treated as originating request for a registered user.
- When the **ignore-psu-sesscase** option is set:
  - The request is a dialog creating request or a standalone request.
  - There is no "odi" parameter in the top route of the request.
  - There is an "orig" parameter in the top route of the request.
  - The served user is registered

Originating request - Unregistered User

When the Oracle USM receives an Initial request, it is validated as an originating request from or on behalf of an unregistered user when the following conditions are met:

- When the **ignore-psu-sesscase** option is not set:
  - The request is a dialog creating request or a standalone request.
  - The served user is unregistered.
  - The request is from an AS or I-CSCF and the top route header contains the orig parameter OR
    The regstate and sescase of the P-served-user header indicates that the request is an originating request for an unregistered user.
- When the **ignore-psu-sesscase** option is set:
  - The request is a dialog creating request or a standalone request.
  - There is no "odi" parameter in the top route of the request.
  - There is an "orig" parameter in the top route of the request.
  - The served user is unregistered

Terminating Requests - Registered User

When the Oracle USM receives an Initial request, it is validated as a terminating request towards a registered user when the following conditions are met:

- When the **ignore-psu-sesscase** option is not set:
  - The request is a dialog creating request or a standalone request.
  - There is no "orig" parameter in the top route of the request.
  - There is no "odi" parameter in the top route of the request.

  - The regstate and sescase parameters of the P-served-user indicate for this to be treated as terminating request for a registered user OR the request is finished with originating services if applicable and the request is destined to a user who is currently registered with the Oracle USM.
  - If the Request-URI changes when visiting an application server, the Oracle USM terminates the checking of filter criteria and routes the request based on the changed value of the Request-URI, per 3GPP Specification TS 23.218.
- When the **ignore-psu-sesscase** option is set:
  - The request is a dialog creating request or a standalone request.
  - There is no "odi" parameter in the top route of the request.
  - There is no "orig" parameter in the top route of the request.
  - The served user is registered

Terminating Requests - Unregistered User

See the IFC Support for Unregistered Users section in the Configuration Guide for this case.

- When the **ignore-psu-sesscase** option is not set:
  - If the Request-URI changes when visiting an application server, the Oracle USM terminates the checking of filter criteria and routes the request based on the changed value of the Request-URI, per 3GPP Specification TS 23.218.
- When the **ignore-psu-sesscase** option is set:
  - The request is a dialog creating request or a standalone request.
  - There is no "odi" parameter in the top route of the request.
  - The served user is not registered.

The request is a dialog creating request or a standalone request.

- There is no "orig" parameter in the top route of the request.
- There is no "odi" parameter in the top route of the request.
- The regstate and sescase parameters of the P-served-user indicate for this to be treated as terminating request for an unregistered user

Third Party Registration for an Implicit Registration Set

When using iFCs, the Oracle USM performs third party registrations based on the iFC downloaded for each PUID. By default, the Oracle USM performs third party registration for the service profiles of all PUID's in a user's implicit registration set. This is compliant with 3GPP specifications. The system includes any shared or default iFCs that apply to each PUID during this process. The system performs this function when it receives user-initiated de-registrations, but not when it receives RTRs. If desired, the user can configure the Oracle USM to perform third party registration for only the REGISTERED PUID in the registration using a **sip-registrar** option.

> **Note:**
>
> The Oracle USM does not attempt third party registration for any barred, tel or wildcard PUIDs.

The user can verify all third party registrations using the **show registration sipd by-user [user] detailed** command. Example output is shown below.

```
ORACLE# show registration sipd by-user 234 detailed

Registration Cache (Detailed View)    Wed Sep 16 2015  10:57:44

User: sip:234@acme-ims.com
  Registered at:  2015-09-16-10:57:40    Surrogate User: false
  Emergency Registration? No
  ContactsPerAor Rejects 0
  ContactsPerAor OverWrites 0

  Contact Information:
    Contact:
      Name: sip:234@acme-ims.com
      Valid: true
      Challenged: false
      Registered at: 2015-09-16-10:57:40
      Last Registered at: 2015-09-16-10:57:40
      Expire: 3596
      Local expire: 296
      Half: 1796

      Registrar IP: 0.0.0.0
      Transport: UDP
      Secure: false
      Local IP: 192.168.53.99:5060

      User Agent Info:
        Contact: sip:234@192.168.53.181:5060
        Realm: core
        IP: 192.168.53.181:5060

      SD Info:
        Contact: sip:234-tbcktcgo177fc@192.168.53.99:5060
Call-ID: 1-5853@192.168.53.181
      Path: <sip:234@192.168.53.181:5060;lr;p-acme-serving>

    Associated URI(s):
      URI: sip:234@acme-ims.com
      Status: Non-Barred
         Filter Criteria:
           Priority: 0
             Filter: ((method == REGISTER)) or
              ((method == INVITE))
             Application Server: sip:172.16.17.10:5060

      URI: sip:1@acme-ims.com
      Status: Non-Barred
         Filter Criteria:
           Priority: 0
             Filter: ((method == REGISTER)) or
              ((method == INVITE))
```

```
            Application Server: sip:172.16.17.10:5060
          Priority: 1
            Filter: ((method == INVITE)) or
             ((method == REGISTER))
            Application Server: sip:172.16.53.181:5065

      URI: tel:135
      Status: Barred
         Filter Criteria:
          Priority: 0
            Filter: ((method == INVITE)) or
             ((method == REGISTER))
            Application Server: sip:172.16.53.181:5065
          Priority: 1
            Filter: ((method == INVITE)) or
((method == REGISTER))
            Application Server: sip:172.16.53.181:5095


    Third Party Registration(s):
      Third Party Registration Host: 172.16.17.10
      Registration State: REGISTERED
      Last Registered at: Never
      Third Party Registration Host: 172.16.53.181
      Registration State: REGISTERED
      Last Registered at: Never
```

The user can check for third party registrations errors using the **show sipd third-party-reg all** command. Example output is shown below.

```
ORACLE# show sipd third-party-reg all
3rd Party Registrar             SA State   Requests     200OK  Timeouts    Errors
(D)111.11.17.10                   INSV         1         1        0         0
(D)111.11.53.181                  INSV         1         1        0         0
```

The user can disable the default behavior and perform third party registration only for the PUID in the REGISTER by configuration. Disabling this behavior can improve system performance by preventing the system from having to walk through large PUID sets for large numbers of ASs. The ACLI syntax for disabling this functionality using the **disable-thirdPartyReg-for-implicit-puid** setting follows.

```
ORACLE(sip-registrar)#options +disable-thirdPartyReg-for-implicit-puid
```

> **Note:**
>
> Prior to this version, the Oracle USM's default behavior was the same as if the **disable-thirdPartyReg-for-implicit-puid** option was set in the SIP registrar. Users upgrading to this version of the Oracle USM must set the **disable-thirdPartyReg-for-implicit-puid** option to retain the previous behavior.

TEL URI Replacement with SIP URI in R-URI to AS

When the USM receives a request containing a TEL URI from the Media Gateway Control Function (MGCF), it sends the TEL URI as an R-URI to the Application Server (AS) to perform services. However, in some implementations, the AS does not accept TEL URI and requires the trigger to be based on SIP URI. This feature, when enabled, causes the USM to replace the TEL R-URI with a SIP URI based on the first SIP user in the implicit set.

In the current implementation of the USM for terminating calls, when the USM receives an R-URI with SIP user=phone (for example, "sip:+359888528650@sip.mtel.bg; user=phone"), the USM replaces the SIP URI with a TEL URI and further uses the TEL URI (for example, "tel: +359888528650") for Location Information Requests (LIR) and Server Assignment Requests (SAR) when the user is not in the registration cache. The Server Assignment Answer (SAA) provides the Public Identity "sip:tel.359888528650@sip.mtel.bg" in the Service Profile as it's part of the implicit registration set and the USM stores it in its registration cache. Then, based on the Service Profile for the TEL URI, the USM triggers the AS using the R-URI "tel: +359888528650". However, in some implementations, for requests coming from the MGCF, the USM receives requests with a TEL URI which is sent as an R-URI to the AS while doing services, but the AS does not accept TEL URIs and requires the trigger to be based on a SIP URI.

To rectify this deficiency, this feature, when activated and when the USM is the assigned Serving Call Session Control Function (S-CSCF), causes the USM to replace the TEL R-URI with the first SIP URI in the implicit set for the TEL user; it then performs services based on the trigger for the user of the first implicit SIP URI. Once a TEL URI is changed to a SIP URI to perform services it will not be changed back to a TEL URI for the entire call flow. When this feature is enabled, the USM uses the first SIP user entry in the implicit set and performs services for the user irrespective of whether the user is in the registration cache.

TEL URI Replacement with SIP URI in R-URI to AS Configuration

Configuration changes occur in real time and do not require rebooting.

1. Access the **ifc-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# ifc-profile
   ORACLE(ifc-profile)#
   ```

2. Select the **ifc-profile** object to edit.

   ```
   ORACLE(ifc-profile)# select
   <name>:
   1:  name=ifc_appserver

   ORACLE(ifc-profile)#
   ```

3. **options** — Set the options parameter by typing **options** , a space, the option name **replace-tel-ruri-with-implicit-sip** with a plus sign in front of it, and then press Enter. You must prepend the new option with a plus sign to append the new option to the IFC profile's options list. If you type the option without the plus sign, you will overwrite any previously configured options.

   ORACLE(ifc-profile)# options +replace-tel-ruri-with-implicit-sip

4. Type **done** to save your configuration.

Additional Options

- The Oracle USM can populate the top Route: header with the sescase value for ASs that require it. In such a case, the parameter is created as either call=orig or call=term. This behavior is enabled by configuring the **add-sescase-to-route** option in the ifc-profile.
- When the dialog-transparency parameter in the sip-config is set to enabled and your network includes multiple ASs, you should add the **dialog-transparency-support** option in the ifc-profile.

- The Oracle USM provides an alternative, configurable option that allows the user to specify the use of route header information to determine Served User and Session Case for out-of-the-blue (OOTB) calls. This method is 3GPP-compliant. By default, the Oracle USM uses information from the P-Served-User (PSU) header. The user configures this behavior by enabling the ignore-psu-sesscase option in the ifc-profile.

IFC Support for Unregistered Users

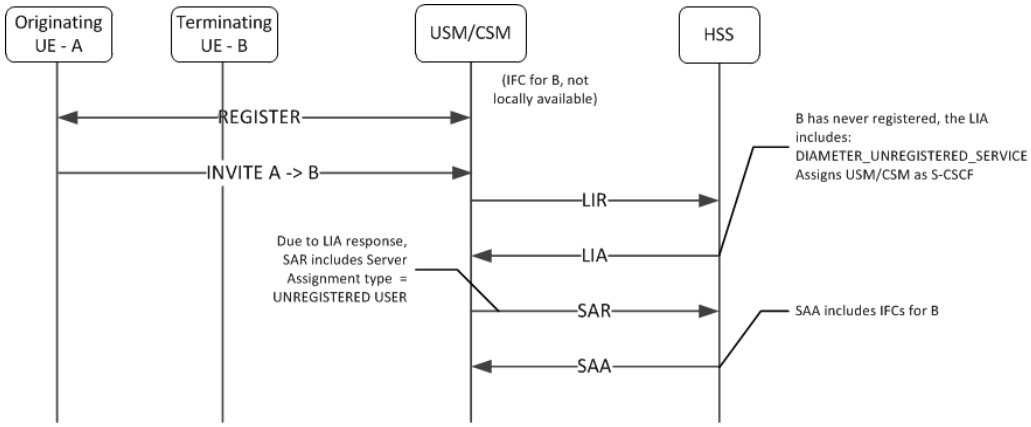
The Oracle USM can download Initial Filter Criteria (IFC) from the HSS for unregistered users. This section displays applicable message sequence diagrams.

UE-terminating requests to an unregistered user

The Oracle USM downloads and executes IFCs for the terminating end of calls. The following call flows indicate possible cases for the terminating unregistered user.

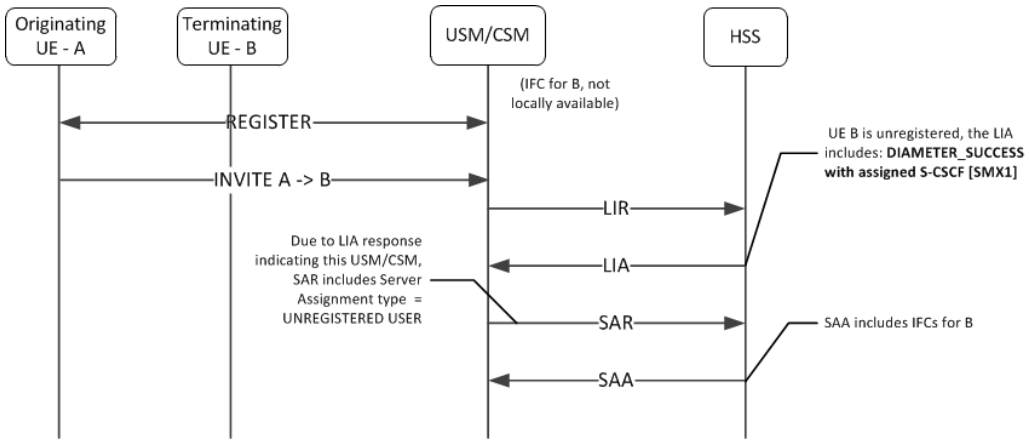
Terminating UA - Unregistered

UE has never registered.

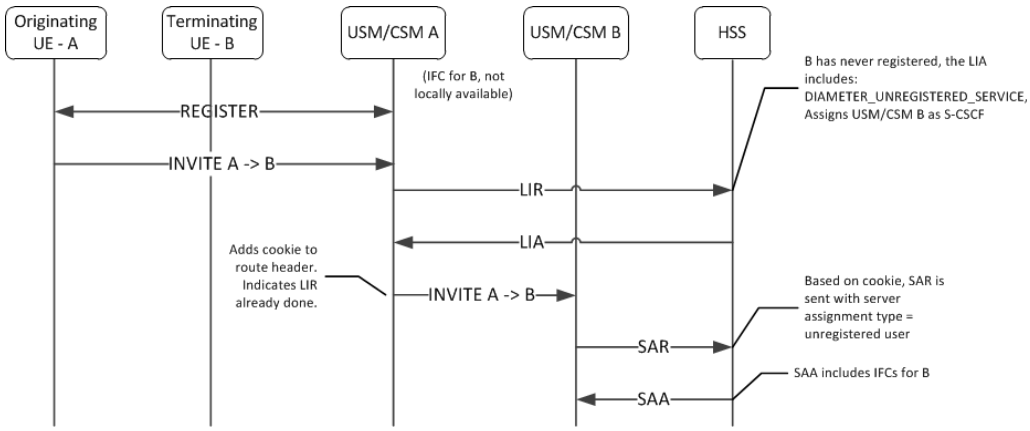


Terminating UA - Unregistered

UE originally registered as a consequence of an originating or terminating request or an S-CSCF has stored the user profile.



Terminating UA - Not Registered, Served by other Oracle USM



UE Subsequent Registration

If the Oracle USM has a cached IFC downloaded for an unregistered UA who later registers to that Oracle USM, the cached IFC will be cleared and updated with the IFC downloaded by the registration process.

Caching the Downloaded IFC

When the Oracle USM downloads IFCs for unregistered users, they are saved to a local cache. If the IFC cache fills up, an older cached IFC for a user is released.

Optimizing IFC Updates

The Oracle USM aims to reduce the number of IFC updates traversing the network to save bandwidth and transactional overhead. Unless the unregistered UE's IFC entry has been deleted because of exhausting cache space, the following optimizations are performed:

- If IFCs are available locally, then an SAR/SAA operation to download IFCs will not be performed.

- If a previous IFC download operation did not return any IFCs, then subsequent calls to that unregistered user will not invoke the SAR/SAA messaging to download IFCs.

Push Profile Request (PPR) updates

The HSS can push service profile updates for private IDs. The Oracle USM can process PPR updates for unregistered entities. If the user entry has been deleted because IFC cache space has been exhausted, the PPRs will not be processed.

ACLI Instructions

SIP Registrar

To create an IFC Profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **ifc-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# ifc-profile
   ORACLE(ifc-profile)#
   ```

4. **name**—Enter a name for this IFC profile.
5. **state**—Set this to enabled to use this ifc-profile.
6. **options**—Set the options parameter by typing options, a Space, the option name with a plus sign in front of it, and then press Enter.

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the options list, you must prepend the new option with a plus sign.

   The options included in this section are: **add-sescase-to-route** and **dialog-transparency-support.**

7. Type **done** when finished.

SIP Registrar

To enable IFC support in a SIP Registrar:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-registrar
ORACLE(sip-registrar)#
```

4. Type **select** and choose the number of the pre-configured SIP registrar configuration element you want to configure.

```
ORACLE(sip-registrar)# select
name:
1: registrar1
selection:1
ORACLE(sip-registrar)#
```

5. **ifc-profile**—Set this parameter to the name of the IFC profile you just created.
6. **serving-function**—Set this parameter to disabled when you want the Oracle USM to act solely as an I-CSCF. When disabled, the Oracle USM always forwards requests from unregistered users to the serving group.
   The default is enabled, which retains the S-CSCF function on this Oracle USM.
7. **serving-group**—Set this parameter to a Session Agent Group (SAG) name. The Oracle USM forwards requests from unregistered users to this group when the serving function parameter is disabled.
   Use of this parameter requires the prior configuration of a SAG that includes all prospective S-CSCFs. The name you give to that group is the name you specify as an argument to this parameter.
8. Type **done** when finished.

Shared and Default iFCs

The Oracle USM supports Shared iFCs (SiFC), as defined by TS 29.229 and Default iFCs, which are an Oracle extension upon SiFCs. SiFCs provide an operator with the ability to create iFC templates and apply them to a large number of UEs. The SiFC process optimizes the provisioning, storage and transfer of service profile information. The default IFC (DiFC) establishes a configuration wherein the iFC associations are available on the Oracle USM itself. This establishes a backup scenario in case the HSS is not responsive.

To support the SiFC feature on the Oracle USM, you create a profile that refers to a local, XML-formatted file. This file specifies the iFCs to be shared. You apply these profiles to registrars to specify where they are used.

When an SiFC configuration is in place, the Oracle USM notifies the HSS that it supports SiFCs within the Supported-Features AVP in the SAR. The HSS replies to confirm that it supports SiFCs within the SAA. The SiFC feature must be enabled on the HSS.

Note that the form and function of the SiFC and DiFC files are compatible. You can use the same file for both SiFC and DiFC configuration, if desired.

SiFC Usage

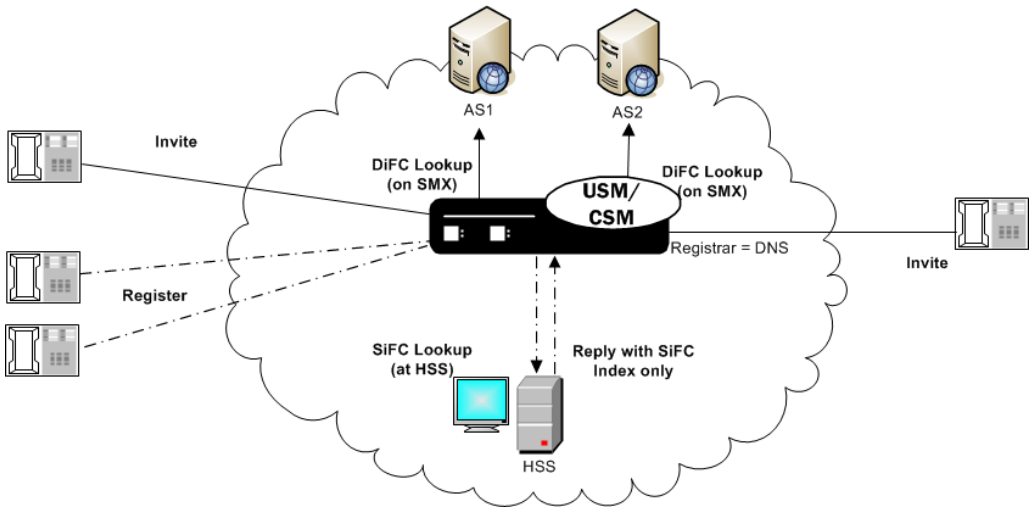
When an applicable end station registers, the Oracle USM forwards the registration to the HSS normally. Given SiFC configuration however, the HSS sends a service-profile containing the SiFC identifiers to the Oracle USM rather than the entire service definition. The Oracle USM parses these identifiers and maps the user to the locally stored filter criteria.

The <IFCSet id =”x”> tags in the XML file on the Oracle USM map to the HSS identifiers.

DiFC Usage

In contrast to SiFCs, the Oracle USM fires DiFCs within the context of a session. During the session, the Oracle USM associates the iFCs within the DiFC file with the user, as needed. DiFC usage is invoked during session initiation.

Note that DiFCs are database agnostic. You can use DiFCs for HSS, ENUM and local database configurations. An operational overview of SiFCs and DiFCs is shown below.



SiFC/DiFC File Example

An example of a Oracle USM local SiFC/DiFC XML file, including a single iFC Set containing a single iFC, is presented below.

```
<?xml version="1.0" encoding="UTF-8"?>
<IFCSets>
  <IFCSet id=”0”>
    <InitialFilterCriteria>
      <Priority>0</Priority>
      <TriggerPoint>
        <ConditionTypeCNF>0</ConditionTypeCNF>
        <SPT>
          <ConditionNegated>0</ConditionNegated>
          <Group>0</Group>
          <Method>INVITE</Method>
          <Extension></Extension>
        </SPT>
      </TriggerPoint>
    <ApplicationServer>
      <ServerName>sip:172.16.101.26:5060</ServerName>
      <DefaultHandling>0</DefaultHandling>
    </ApplicationServer>
    <ProfilePartIndicator>0</ProfilePartIndicator>
  </InitialFilterCriteria>
</IFCSet>
</IFCSets>
```

Note that the Shared IFCSet contains the integer value property (id=”0”) that associates these filter criteria with users registered with the Oracle USM. In the case of SiFC, it is the value that the HSS should send when referencing shared sets. In the case of DiFC, the integer is meaningless. The Oracle USM loads and executes default iFCs in the order they appear within the XML file.

iFC Execution Order

Within the context of the 3GPP standards, the Oracle USM evaluates explicitly downloaded iFCs first when determining where to look for a service. If the Oracle USM cannot execute on the service based on explicitly downloaded iFCs, it refers to the SiFC, then the DiFC information to identify an AS that supports the service.

Refreshing SiFC and DiFC Files

Given the nature of local file usage, an ACLI command is available to allow the user to refresh SiFC and DiFC contexts in memory after the user has saved changes to the SiFC and DiFC files. Run the following command to deploy these changes:

ORACLE# **refresh ifc <ifc-profile name>**

Note also that the Oracle USM validates the SiFC and DiFC files whenever you Activate your configuration.

SiFC and DiFC Configuration

To configure the Oracle USM to use Shared and Default IFCs:

1. From superuser mode, use the following command sequence to access ifc-profile element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# ifc-profile
   ```

2. Define your profile.
3. **name**—Enter a name for this IFC profile.

   ```
   ORACLE(ifc-profile)# name acmeTelecomIFC
   ```

4. **state**—Set this to enabled to use this ifc-profile.

   ```
   ORACLE(ifc-profile)# state enabled
   ```

5. **default-ifc-filename**—Specify filename and, if not stored in the default directory /code/ifc, the applicable pathname.

   ```
   ORACLE(ifc-profile)# default-ifc-filename Afile.xml.gz
   ```

6. **shared-ifc-filename**—Specify filename and, if not stored in the default directory /code/ifc, the applicable pathname.

   ```
   ORACLE(ifc-profile)# shared-ifc-filename Bfile.xml.gz
   ```

7. **options**—Set the options parameter by typing options, a Space, the option name with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(ifc-profile)# done
   ```

8. Apply the ifc-profile to your sip registrar.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-registrar
```

Select the registrar you want to configure and apply the profile.

```
ORACLE(sip-registrar)# select
ORACLE(sip-registrar)# ifc-profile acmeTelecomIFC
ORACLE(sip-registrar)# done
```

Distinct and Wildcarded Public Service Identity (PSI) Support

The Oracle USM supports the use of distinct Public Service Identity (PSI) and wildcarded PSIs, typically for specifying access to a service. There is no configuration required on the Oracle USM to enable this support.

Administrators use individual PSI entries and/or wildcarded PSIs as service identifiers on an HSS. These identifiers provide the information needed to direct applicable messages to applicable application servers. Distinct PSIs can reside within individual PSI entries; wildcarded PSI entries are managed within iFC lists. Wildcarded PSI support is typically implemented to reduce HSS resource requirements. By configuring a wildcarded PSI, administrators can use a single record within the iFC to manage multiple resources.

A wildcard is composed of an expression that, when used in a user part, provides for access to multiple service resources. The regular expressions themselves are in form of Perl Compatible Extended Regular Expressions (PCRE).

For example, consider the following two service resources:

- sip:chatroom-12@core.com
- sip:chatroom-64@core.com

These two service resources can be represented simultaneously at the HSS using the following syntax:

- sip:chatroom-!.*!@core.com

The Oracle USM caches filter criteria information that uses this wildcard syntax. This avoids the need for SAR/SAA exchanges between the Oracle USM and the HSS every time an entity requests the service. The Oracle USM is equally capable of caching distinct PSIs, which similarly offloads the need for SAR/SAA exchanges during service resource location processes.

For most call flows, the Oracle USM does not evaluate the expression for the purpose of finding a match. Instead, it keeps the syntax provided by the HSS in its cache and provides the wildcarded syntax in the applicable AVP.

To allow the feature, the Oracle USM supports:

- Wildcarded public user identity AVP in the LIA, SAR and SAA
- User Profile AVP in the SAA
- P-Profile-Key across the Mw interface, as defined in RFC 5002

Configuring SIP Ping OPTIONS Support

You can configure the Oracle USM to respond to SIP ping OPTIONS. This support is typically configured on an S-CSCF so it can respond to pings OPTIONS sent by a P-CSCF:

To configure an SIP Options Ping response support:

1. From superuser mode, use the following command sequence to access ping-response command on a sip-interface element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)# sel
   ```

2. Enable the support with the ping-response command.

   ```
   ORACLE(http-config)# ping-response enabled
   ORACLE(http-config)# done
   ```

   ping-response—Enable ping-response to allow your device to respond to ping OPTIONS. For example, this feature is useful within hybrid deployment environments on a P-CSCF as a means of verifying the S-CSCF's availability. This configuration allows the S-CSCF to respond to SIP ping OPTIONS.

Redundancy and Load Balancing with HSS Servers

The Oracle USM allows you to operate with multiple HSS servers, supporting:

- Redundancy - Continue normal operation despite HSS failure.
- Load Balancing - Divide the traffic load between HSS servers in a group of HSSs. Preference is based on the HSS list order configured on the Oracle USM.

You configure HSS servers within HSS Groups to support this functionality. For redundancy, you create and assign HSS groups, and apply either the hunt or fail-over strategy to your HSS group. To implement load balancing, you configure the applicable HSS group with a the round-robin server allocation strategy. This functionality assumes the HSS infrastructure itself is configured for redundancy.

The Oracle USM establishes and manages multiple Cx connections with each applicable HSS. This management is achieved by connection identifiers on the Oracle USM that allow it to distinguish between connections. This provides the network with the flexibility of being able to use multiple paths to a given HSS regardless of AVP values.

About HSS Groups

You configure HSS groups based on your redundancy and failover design. You accomplish this by configuring your HSS groups with the applicable HSS servers. You then assign your group to a registrar. HSS group configuration does not preclude assigning an HSS in the group to a registrar individually.

HSS groups can contain individual HSSs. Members of an HSS group are prioritized by the server list; the first server in the list takes the highest priority; the last takes the lowest. You can manually disable an HSS group, if desired, which prevents the Oracle USM from attempting to access any of the HSS servers via that group.

HSS group members do not need to reside in the same domain, network, or realm. The Oracle USM can allocate traffic among member HSSs regardless of their location. It uses the allocation strategies you configure for the group to distribute traffic across the group members.

Group allocation strategies define how the Oracle USM selects an HSS. For example, the hunt strategy selects HSSs in the order in which they are listed. Allocation strategies include the following:

Allocation Strategy	Description
failover	For HSS redundancy deployments, the failover strategy specifies that the Oracle USM selects the next highest priority HSS server for all operations if the first HSS fails. The Oracle USM does not resume operation with the initial HSS when it comes back into service.
hunt	For HSS redundancy deployments, the hunt strategy specifies that the Oracle USM select HSSs in the order in which they are configured in the HSS group. If the first HSS is available, all traffic is sent to the first HSS. If the first HSS is unavailable, all traffic is sent to the second HSS. The system follows this process for all HSS servers in the group. When a higher priority HSS returns to service, all traffic is routed back to it.
roundrobin	This strategy targets HSS load balancing deployments. The Oracle USM selects each HSS in the order in which it appears in the group list, routing diameter requests to each HSS in turn.

Paths taken by specific messaging is constrained by the purpose of that messaging, and refined by a group's allocation strategy. Applicable messaging includes UAR/UAA, MAR/MAA, SAR/SAA and LIR/LIA. For both failover and hunt strategies, all messaging is sent to the current active server. For the round-robin strategy, messaging is distributed to group members sequentially, using the member list order.

Connection Failure Detection

The Oracle USM detects that a connection between itself and a given HSS has failed if either a diameter request fails or the diameter DWR/DWA handshake fails. If the HSS does not respond to five requests, the Oracle USM marks that HSS as out of service.

The Oracle USM forwards unacknowledged messages to subsequent HSSs based on strategy. It changes the destination host AVP of these messages and marks then with the T flag. The HSS recognizes the T flag as an indication that the request may be a duplicate, caused by a problem in the network.

Periodically, the Oracle USM attempts to establish diameter connections with out of service HSS servers. When those connections succeed, the Oracle USM marks the HSS as in-service and resumes using it within the context of the configured redundancy and load balancing strategy.

Configuring HSS Groups

To configure HSS groups:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **hss-group** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# hss-group
   ORACLE(hss-group)#
   ```

4. **name**—Enter a unique name for the HSS group in Name format.
5. **state**—Enable or disable the HSS group on the Oracle USM. The default value is **enabled**. Valid values are:
   - enabled | disabled
6. **origin-host-identifier**— Set this to a string for use in constructing a unique Origin Host AVP. This setting always takes precedence over the origin-host-identifier configured for the home-subscriber-server. This setting is required.
7. **strategy**—Indicate the HSS server allocation strategy you want to use. The strategy you chose selects the HSS servers that will be made available by this hss-group. The default value is **hunt**. The valid values are:
   - **hunt**—Selects HSS servers in the order in which they are listed. For example, if the first server is online, all traffic is sent to the first server. If the first server is offline, the second server is selected. If the first and second servers are offline, the third server is selected. When the Oracle USM detects that a higher priority HSS is back in service, it routes all subsequent traffic to that HSS.
   - **roundrobin**—Selects each HSS server in the order in which they are listed in the destination list, selecting each server in turn, one per session.
   - **failover** — Selects the first server in the list until failure is detected. Subsequent signaling goes to the next server in the list.
8. **hss-configs**—Identify the HSS servers available for use by this hss-group. This list can contain as many HSS servers as is necessary. An hss-config list value must correspond to a valid hss-config.

   Display syntax for the hss-configs parameter by typing the question mark character after the parameter name on the ACLI.

   ```
   ORACLE(hss-group)# hss-configs ?
   <string> list of home-subscriber-server configs for this group
            for single-entry: hss1
            for multi-entry: (hss1 hss2)
            for adding an entry to an existing list: +hss3
            for deleting an entry from an existing list: -hss3
            for multiple entries add/remove from a list: +/-(hss1 hss2)
   ```

   The following example shows an HSS group using the hunt allocation strategy applied.

   ```
   hss-group
           name                          group-test1
           state                         enabled
           origin-host-identifier
           strategy                      hunt
           hss-configs                   hss1, hss2
           last-modified-by              admin@console
           last-modified-date            2013-05-13 14:58:01
   ```

6

Routing with Local Policy

This chapter explains how to configure session routing and load balancing for SIP services using local policy and session agents. It contains information about configuring session agents and session agent groups, as well as local policies that can be used for routing SIP messaging.

Session Agents Session Groups and Local Policy

When you configure session routing for SIP, you can use session agents, session agent groups and local policies to define routing. (Using session agents and session agent groups is not required.)

- session agent: defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes.
- session agent group (SAG): contains individual session agents. Members of a SAG are logically equivalent (although they might vary in their individual constraints) and can be used interchangeably.
  You apply an allocation strategy to the SAG to allocate traffic across the group members. Session agent groups also assist in load balancing among session agents.
- local policy: indicates where session request messages, such as SIP INVITES, are routed and/or forwarded. You use a local policy to set a preference for selecting one route over another.

Another element of routing is the realm. Realms are used when an SBC communicates with multiple network elements over a shared intermediate connection. Defining realms allows sessions to go through a connection point between the two networks.

When you configure a realm, you give it an identifier, which stores the name of the realm associated with a sip-interface. The realm identifier value is also needed when you configure session agents and local policies. You can associate a realm with a session agent to identify the realm for sessions coming from or going to the session agent. You also need the realm identifier when you configure local policy to identify the egress realm (realm of the next hop).

SIP Session Agents

SIP session agents can include the following:

- softswitches
- SIP proxies
- application servers
- SIP gateways
- SIP endpoints

In addition to functioning as a single logical next hop for a signaling message (for example, where a SIP INVITE is forwarded), session agents can provide information about next or previous hops for packets in a SIP agent, including providing a list of equivalent next hops.

You can use the session agent to describe one or more SIP next or previous hops. Through the configured carriers list, you can identify the preferred carriers to use for traffic coming from the session agent. This set of carriers will be matched against the local policy for requests coming from the session agent. You can also set constraints for specific hops.

Session Agent Status Based on SIP Response

The Oracle USM can take session agents out of service based on SIP response codes that you configure, and you can also configure SIP response codes that will keep the session agent in service.

With this feature disabled, the Oracle USM determines session agents' health by sending them ping messages using a SIP method that you configure. Commonly, the method is an OPTIONS request. If it receives any response from the session agent, then the Oracle USM deems that session agent available for use.

However, issues can arise when session agents are administratively out of service, but able to respond to OPTIONs requests. A session agent like this might only respond with a 200 OK when in service, and send a 4xx or 5xx message otherwise.

The session agent status feature lets you set the SIP response message that either takes a session agent out of service or allows it to remain in service when it responds to the Oracle USM's ping request.

Details of this feature are as follows:

- The Oracle USM only considers a session agent in service when it responds to a request method you set with the final response code that you also set. If a final response code is set, then provisional responses are not used for determining whether or not to take a session agent out of service. If the Oracle USM receives a final response code that does not match the session agent configuration, it treats the session agent as though it had not responded.
- The Oracle USM takes a session agent out of service when it receives an error response for dialog creating request with a response code listed in the new **out-service-response-codes** parameter.

In the case where a the session agent's response has a Retry-After header, the Oracle USM tries to bring the session agent back into service after the period of time specified in the header. To do so, it sends another ping request.

There are two lists you can configure in the session agent configuration to determine status:

- In-service list—Set in the ACLI **ping-in-service-response-codes** parameter, this list defines the response codes that keep a session agent in service when they appear in its response to the Oracle USM's ping request. Furthermore, the Oracle USM takes the session agent out of service should a response code be used that does not appear on this list.
- Out-of-service list—Set in the ACLI **out-service-response-codes** parameter, this list defines the response codes that take a session agent out of service when they appear in its response to the Oracle USM's ping request or any dialog-creating request.

When the Oracle USM receives a session agent's response to its ping request, it first checks to see if there is an in-service list of responses configured for that session agent. If the list is configured and the Oracle USM determines that there is a match, the session agent is deemed in service. Otherwise it takes the session agent out of service. In this way, the in-service list takes precedence over the out-of-service list. If you configure the in-service list, then the Oracle USM ignores the out-of-service list.

If there is no list of in-service responses for the session agent, then the Oracle USM checks the out of service list. If it is configured and the Oracle USM determines that there is a match, the Oracle USM removes that session agent from service. If there is no match, then the session agent is deemed in service.

SIP Session Agent Continuous Ping

You can configure the Oracle USM to use either a keep-alive or continuous method for pinging SIP session agents to determine their health—i.e., whether or not the Oracle USM should route requests to them. To summarize the two methods:

- keep-alive— Oracle USM sends a ping message of a type you configure to the session agent in the absence of regular traffic. Available in Release C5.1.0 and in earlier releases.
- continuous—The Oracle USM sends a ping message regardless of traffic state (regular or irregular); the Oracle USM regularly sends a ping sent based on the configured ping interval timer. Available in Release C5.1.1p6 and in later releases.

By sending ping messages, the Oracle USM monitors session agents' health and can determine whether or not to take a session out of service (OOS), leave it in service, or bring it back into service after being OOS.

When you set it to use the keep-alive mode of pinging (available in Release C5.1.0 and before), the Oracle USM starts sending a configured ping message to a session agent when traffic for that session agent has become irregular. The Oracle USM only sends the ping if there are no SIP transactions with a session agent over a configurable period of time, to which the session agent's response can have one of the following results:

- Successful response—A successful response is either any SIP response code or any response code not found in the **out-service-response-codes** parameter; these leave the session agent in service. In addition, any successful response or any response in the **ping-in-service-response-codes** parameter can bring a session agent from OOS to in-service status.
- Unsuccessful response—An unsuccessful response is any SIP response code configured in the **out-service-response-codes** parameter and takes the session agent sending it OOS. Because this parameter is blank by default, the Oracle USM considers any SIP response code successful.
- Transaction timeout—A transaction timeout happens when the session agent fails to send a response to the Oracle USM's request, resulting in the session agent's being taken OOS.

Despite the fact that the keep-alive ping mode is a powerful tool for monitoring session agents' health, you might want to use the continuous ping method if you are concerned about the Oracle USM not distinguishing between unsuccessful responses from next-hop session agents and ones from devices downstream from the next-hop session agent. For example, if a SIP hop beyond the session agent responds with a 503 Service Unavailable, the Oracle USM does not detect whether a session agent or the device beyond it generated the response.

When you use the continuous ping method, only the next-hop session agent responds—preventing the request from being sent to downstream devices. The Oracle USM also sends the ping in regular traffic conditions when in continuous ping mode, so it is certain the response comes from the next hop associated with the session agent. And in continuous ping mode, only entries for the **ping-out-service-response-codes** parameter and transaction timeouts bring session agents OOS.

SIP SA Continuous Ping Configuration

You can set the ping mode in the session agent or session constraints configuration. For backward compatibility, the default for the ping-send-mode parameter is keep-alive, or the functionality available in Release C5.1.0 and in earlier releases.

To configure the ping mode for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

4. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).

5. Save and activate your configuration.

   To configure the ping mode for the session constraints:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

8. Type **session-constraints** and press Enter.

   ```
   ORACLE(session-router)# session-constraints
   ORACLE(session-constraints)#
   ```

   If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

9. **ping-send-mode**—If to want to use continuous ping mode to send ping messages to session agents in regular traffic conditions, set this parameter to **continuous**. If you want to use the keep-alive mode, leave this parameter set to **keep-alive** (default).

10. Save and activate your configuration.

About Session Agents

This section describes session agents. A session agent defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes. Service elements

such as gateways, softswitches, and gatekeepers are defined automatically within the Oracle USM as session agents. For each session agent, concurrent session capacity and rate attributes can be defined. You can group session agents together into session agent groups and apply allocation strategies to achieve traffic load balancing.

You can assign a media profile to a session agent.

You can configure a set of attributes and constraints for each session agent to support the following:

- session access control: Oracle USM only accepts requests from configured session agents
- session admission control (concurrent sessions): Oracle USM limits the number of concurrent inbound and outbound sessions for any known service element.
- session agent load balancing: session agents are loaded based on their capacity and the allocation strategy specified in the session agent group.
- session (call) gapping: Oracle USM polices the rate of session attempts to send to and receive from a specific session agent.

Session Agent Groups

Session agent groups contain individual session agents. Members of a session agent group are logically equivalent (although they might vary in their individual constraints) and can be used interchangeably. You can apply allocation strategies to session agent groups.

Examples of session agent groups include the following:

- application server cluster
- media gateway cluster
- softswitch redundant pair
- SIP proxy redundant pair
- gatekeeper redundant pair

Session agent group members do not need to reside in the same domain, network, or realm. The Oracle USM can allocate traffic among member session agents regardless of their location. It uses the allocation strategies configured for a SAG to allocate traffic across the group members.

Allocation strategies include the following:

Allocation Strategy	Description
Hunt	Oracle USM selects the session agents in the order in which they are configured in the SAG. If the first agent is available, and has not exceeded any defined constraints, all traffic is sent to the first agent. If the first agent is unavailable, or is in violation of constraints, all traffic is sent to the second agent. And so on for all session agents in the SAG. When the first agent returns to service, the traffic is routed back to it.
Round robin	Oracle USM selects each session agent in the order in which it is configured, routing a session to each session agent in turn.
Least busy	Oracle USM selects the session agent with the least number of active sessions, relative to the maximum outbound sessions or maximum sessions constraints (lowest percent busy) of the session agent.
Proportional distribution	Session agents are loaded proportionately based upon the respective maximum session constraint value configured for each session agent.

Allocation Strategy	Description
Lowest sustained rate	Oracle USM routes traffic to the session agent with the lowest sustained session rate, based on observed sustained session rate.

You apply allocation strategies to select which of the session agents that belong to the group should be used. For example, if you apply the Hunt strategy session agents are selected in the order in which they are listed.

Request URI Construction as Sent to SAG-member Session Agent

The Oracle USM constructs the request URI for a session agent selected from a session agent group by using the **session-agent** > **hostname** value of the selected **session-agent** target. This default behavior enables features such as trunk groups and ENUM to work properly. However, care must be given when the **hostname** parameter is not a resolvable FQDN. The **sag-target-uri**=<value> option can be used to overcome the default behavior.

The value is either

- **ip** – request URI constructed from **session-agent** > **ip-address**
- **host** – request URI constructed from **session-agent** > **hostname**

This option is global and is configured in the **sip-config** configuration element.

Request URI Construction as Forwarded to SAG-member Session Agent

The Oracle USM constructs the request URI for a session agent selected from a session agent group by using the **session-agent** > **hostname** value of the selected **session-agent** target. This default behavior enables features such as trunk groups and ENUM to work properly. However, care must be given when the **hostname** parameter is not a resolvable FQDN. Use the **sag-target-uri**=<value> option to override the default behavior.

The value can be set to either:

- **ip** - request URI constructed from **session-agent** > **ip-address**
- **host** - request URI constructed from **session-agent** > **hostname**

This option is global and is configured in the **sip-config** configuration element.

SIP Session Agent Group Recursion

You can configure a SIP session agent group (SAG) to try all of its session agents rather than to the next-best local policy match if the first session agent in the SAG fails.

With this feature disabled, the Oracle USM performs routing by using local policies, trunk group URIs, cached services routes, and local route tables. Local policies and trunk group URIs can use SAGs to find the most appropriate next-hop session agent based on the load balancing scheme you choose for that SAG: round robin, hunt, proportional distribution, least busy, and lowest sustained rate. When it locates a SAG and selects a specific session agent, the Oracle USM tries only that single session agent. Instead of trying other members of the SAG, the Oracle USM recurses to the local policy that is the next best match. This happens because the Oracle USM typically chooses a SAG based on the fact that it has not breached its constraints, but the Oracle USM only detects failed call attempts (due to unreachable next hops, unresolved

ENUM queries, or SIP 4xx/5xx/6xx failure responses) after it has checked constraints. So the Oracle USM only re-routes if there are additional matching local policies.

When you enable SIP SAG recursion, the Oracle USM will try the additional session agents in the selected SAG if the previous session agent fails. You can also set specific response codes in the SAG configuration that terminate the recursion. This method of terminating recursion is similar to the Oracle USM's ability to stop recursion for SIP interfaces and session agents.

Session agents are selected according to the strategy you set for the SAG, and these affect the way that the Oracle USM selects session agents when this feature enabled:

- Round robin and hunt—The Oracle USM selects the first session agent according to the strategy, and it selects subsequent session agents based on the order they are entered into the configuration.
- Proportional distribution, least busy, and lowest sustained rate—The Oracle USM selects session agents based on the list of session agents sorted by the criteria specified.

You can terminate recursion based on SIP response codes that you enter into the SAG configuration. You can configure a SAG with any SIP response code in the 3xx, 4xx, and 5xx groups. Since you can also set such a list in the session agent configuration, this list is additive to that one so that you can define additional codes for a session agent group with out having to repeat the ones set for a session agent.

About Local Policy

This section explains the role of local policy. Local policy lets you indicate where session requests, such as SIP INVITES, should be routed and/or forwarded. You use a local policy to set a preference for selecting one route over another. The local policy contains the following information that affects the routing of the SIP signaling messages:

- information in the From header
  Information in the message's From header is matched against the entries in the local policy's from address parameter to determine if the local policy applies.
- list of configured realms
  This list identifies from what realm traffic is coming and is used for routing by ingress realm. The source realms identified in the list must correspond to the valid realm IDs you have already configured
- local policy attributes
  The attributes serve as an expression of preference, a means of selecting one route over another. They contain information such as the next signaling address to use (next hop) or whether you want to select the next hop by codec, the realm of the next hop, and the application protocol to use when sending a message to the next hop. You can also use the attributes to filter specific types of traffic.

Routing Calls by Matching Digits

Local policy routing of a call can be based on matching a sequence of digits against what is defined in the local policy. This sequence refers to the first digits in the (phone) number, matching left to right.

The following examples show how the Oracle USM matches an area code or number code against configured local policies.

- If the number or area code being matched is 1234567 (where 123 is an area code), and the from address value in one local policy is 123, and the from address value in another local

policy is 12, the Oracle USM forwards the call to the server that is defined as the next hop in the local policy with 123 as the from address value.

- If the number or area code being matched is 21234, and the from address value in one local policy is 123, and the from address value in another local policy is 12, the Oracle USM will not find a match to either local policy because the first character of the number or area code must match the first character in a from address or to address field.

The following examples show how the Oracle USM matches an area or number code against different local policies: the first one has a From address value of 12 and the second has a From address value of 123. The Oracle USM chooses the route of the local policy that is configured with the most digits matching the area or number code in its From address and To address fields.

- When the two different local policies route to two different servers, and the area or number code being matched is 123, the Oracle USM selects the second local policy based on the From address value of 123.
- When the two different local policies route to two different servers, and the area or number code being matched is 124, the Oracle USM selects the first local policy based on the From address value of 12.

Route Preference

The Oracle USM builds a list of possible routes based on the source realm and the From-address (From-URI) and To-address (Request-URI), which forms a subset from which preference then decides. Any local policy routes currently outside of the configured time/day are not used, if time/day are set. Also, any local policy routes not on the list of carriers (if carriers is set and the requests has a Carrier header) are not used.

> **Note:**
>
> Source realm is used in the local policy lookup process, but it is not used in route preference calculations.

The Oracle USM applies preference to configured local policies in the following order:

1. Cost (cost in local policy attributes) is always given preference.
2. Matching media codec (media profiles option in local policy attributes).
3. Longest matching To address (to address list in local policy).
4. Shortest matching To address (to address list in local policy).
5. Longest matching From address (from address list in local policy).
6. Shortest matching From address (from address list in local policy).
7. Narrowest/strictest day of week specification (days of week option in local policy attributes).
8. Narrowest/strictest time of day specification (start time and end time options in local policy attributes).
9. Wildcard matches (use of an asterisk as a wildcard value for the from address and to address lists in local policy).

10. Wild card matches are given the least preference. A prefix value of 6 is given a higher preference than a prefix value of * even though both prefix values are, in theory, the same length.

DTMF-Style URI Routing

The Oracle USM supports the alphanumeric characters a-d, A-D, the asterisk (*), and the ampersand (#) for local policy matching purposes. The Oracle USM handles these characters as standards DN (POTS) or FQDN when found in the to-addr (req-uri username) or from-addr (from0uri username for SIP, SIPS, and TEL URIs.

In addition, before performing the lookup match, the Oracle USM strips characters that provide ease-of-reading separation. For example, if the Oracle USM were to receive a req-uri containing tel:a-#1-781-328-5555, it would treat it as tel:a#17813285555.

SIP Routing

This section describes SIP session routing. When routing SIP call requests, the Oracle USM communicates with other SIP entities, such as SIP user devices, other SIP proxies, and so on, to decide what SIP-based network resource each session should visit next. The Oracle USM processes SIP call requests and forwards the requests to the destination endpoints to establish, maintain, and terminate real-time multimedia sessions.

Certain items in the messages are matched with the content of the local policy, within constraints set by the previous hop session agent, and the SIP configuration information (for example, carrier preferences) to determine a set of applicable next hop destinations.

The sending session agent is validated as either a configured session agent or a valid entry in a user cache. If the session INVITATION does not match any registering user, the SIP proxy determines the destination for routing the session INVITATION.

Limiting Route Selection Options for SIP

You can configure the local policy to use the single most-preferred route. And you can configure the SIP configuration max routes option to restrict the number of routes which can be selected from a local policy lookup:

- A **max-routes=1** value limits the Oracle USM to only trying the first route from the list of available preferred routes.
- A **max-routes=0** value or no **max-routes** value configured in the options field allows the Oracle USM to use all of the routes available to it.

About Loose Routing

According to RFC 3261, a proxy is loose routing if it follows the procedures defined in the specification for processing of the Route header field. These procedures separate the destination of the request (present in the Request-URI) from the set of proxies that need to be visited along the way (present in the Route header field).

When the SIP NAT's route home proxy field is set to enabled, the Oracle USM looks for a session agent that matches the home proxy address and checks the loose routing field value. If the loose routing is:

- **enabled**—A Route header is included in the outgoing request in accordance with RFC 3261.
- **disabled**—A Route header is not included in the outgoing request; in accordance with the route processing rules described in RFC 2543 (referred to as strict routing). That rule caused proxies to destroy the contents of the Request-URI when a Route header field was present.

Whether loose routing field is enabled is also checked when a local policy ‘s next hop value matches a session agent. Matching occurs if the hostname or the session agent’s IP address field value corresponds to the next hop value. If loose routing is enabled for the matching session agent, the outgoing request retains the original Request-URI and Route header with the next hop address.

About the Ingress Realm

You can create a list of realms in your local policy that is used by the Oracle USM to determine how to route traffic. This list determines from which realm traffic is coming and is used for routing by ingress realm.

The source realm values must correspond to valid identifier entered when the realm was configured.

About the Egress Realm

An egress realm allows SIP signaling to travel out of the Oracle USM through a network other than the home realm. The Oracle USM uses egress realms for signaling purposes (when matching flows). When a packet arrives at the Oracle USM with a destination address that does not match any defined session agents, the Oracle USM uses the address associated with the realm that is, in turn, associated with the SIP configuration’s egress realm ID, as the outgoing network. With the use of the egress realm ID, it is possible to define a default route for SIP requests addressed to destinations outside the home realm. If no egress realm is defined, the home realm (default ingress realm) is used as the default egress realm.

With session agent egress realm configured, the Oracle USM adds a default egress realm to the session agent to identify the signaling interface used for ping requests. The Oracle USM also uses the default egress realm when the normal routing request does not yield an egress realm—for example, when a local policy does not specify the next hop’s realm.

When you configure session agents, you can define them without realms or you can wildcard the realm value. These are global session agents, and multiple signaling interfaces can reach them. Then, when you use session agent pinging, the Oracle USM sends out ping requests using the signaling interface of the default egress realm defined in the global SIP configuration. The global session agents in certain environments can cause problems when multiple global session agents residing in multiple networks, some of which might not be reachable using the default SIP interface egress realm.

The Oracle USM uses the session agent egress realm for ping messages even when the session agent has a realm defined. For normal request routing, the Oracle USM uses the egress realm for global session agents when local policies or SIP-NAT bridge configurations do not point to an egress realm.

Ping Message Egress Realm Precedence

For ping messages, the egress realm precedence occurs in the following way (in order of precedence):

- Egress realm identified for the session agent.
- Session agent realm (set in the realm-id parameter) or the wildcarded value
- Global SIP configuration egress realm, when configured in the egress-realm parameter
- Global SIP configuration home realm

Normal Request Egress Realm Precedence

For normal request routing, the egress realm precedence occurs in the following way (in order of precedence):

- Egress SIP-NAT realm, when the **route-home-proxy** parameter is set to **forced** and no local policy match is found
- Matching local policy realm, when configured in the local policy attributes
- Session agent realm (set in the realm-id parameter) or the wildcarded value
- Session agent egress realm, when configured in the egress-realm-id parameter
- Global SIP configuration egress realm, when configured in the egress-realm parameter
- Global SIP configuration home realm

Session Agent Egress Realm Configuration

Configuring a session agent egress realm is optional.

To configure a session agent egress realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding this feature to an existing configuration, you need to select the configuration (using the ACLI **select** command) before making your changes.

4. **egress-realm-id**—Enter the name of the realm you want defined as the default egress realm used for ping messages. The Oracle USM will also use this realm when it cannot determine the egress realm from normal routing. There is no default value for this parameter.
5. Save and activate your configuration.

About SIP Redirect

SIP redirect involves proxy redirect and tunnel redirect.

Proxy Redirect

You can configure the SIP proxy mode to define how the SIP proxy will forward requests coming from the session agent. This value is used if the session agent's proxy mode has no value (is empty).

Tunnel Redirect

You can use tunnel redirect when requests are routed to a server behind a SIP NAT that sends redirect responses with addresses that should not be modified by the SIP NAT function. For example, a provider might wish to redirect certain calls (like 911) to a gateway that is local to a the UA that sent the request. Since the gateway address is local to the realm of the UA, it should not be modified by the SIP NAT of the server's realm. Note that the server must have a session agent configured with the redirect-action field set to the proxy option in order to cause the redirect response to be sent back to the UA.

SIP Method Matching and To Header Use for Local Policies

For SIP, this feature grants you greater flexibility when using local policies and has two aspects: basing local policy routing decisions on one or more SIP methods you configure and enabling the Oracle USM to use the TO header in REGISTER messages for routing REGISTER requests.

SIP Methods for Local Policies

This feature allows the Oracle USM to include SIP methods in routing decisions. If you want to use this feature, you set a list of one or more SIP methods in the local policy attributes. These are the SIP methods you can enter in the list: INVITE, REGISTER, PRACK, OPTIONS, INFO, SUBSCRIBE, NOTIFY, REFER, UPDATE, MESSAGE, and PUBLISH.

After the Oracle USM performs a local policy look-up for SIP, it then searches for local policy attributes that have this methods list configured. If it finds a a set of policy attributes that matches a method that matches the traffic it is routing, the Oracle USM uses that set of policy attributes. This means that the Oracle USM considers first any policy attributes with methods configured before it considers those that do not have methods. In the absence of any policy attributes with methods, the Oracle USM uses the remaining ones for matching.

In cases where it finds neither matching policy attributes with methods or matching policy attributes without them, the Oracle USM either rejects the calls with a 404 No Routes Found (if the request calls for a response) or drops the call.

You configure local policy matching with SIP methods in the local policy attributes parameter calls **methods**. This parameter is a list that takes either one or multiple values. If you want to enter multiple values, you put them in the same command line entry, enclosed in quotation marks and separated by spaces.

To configure SIP methods for local policy matching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

   ```
   ORACLE(local-policy))# policy-attributes
   ORACLE(policy-attributes)#
   ```

5. **methods**—Enter the SIP methods you want to use for matching this set of policy attributes. Your list can include: INVITE, REGISTER, PRACK, OPTIONS, INFO, SUBSCRIBE, NOTIFY, REFER, UPDATE, MESSAGE, and PUBLISH.

   By default, this parameter is empty—meaning that SIP methods will not be taken into consideration for routing based on this set of policy attributes.

   If you want to enter more than one method, you entry will resemble the following example.

   ```
   ACMEPACKET(local-policy-attributes)# methods "PRACK INFO REFER"
   ```

6. Save and activate your configuration.

Routing Using the TO Header

For the Oracle USM's global SIP configuration, you can enable the use of an ENUM query to return the SIP URI of the Registrar for a SIP REGISTER message. Without this feature enabled, the Oracle USM uses the REQUEST URI. This ability can be helpful because REGISTER messages only have the domain in the REQUEST URI, whereas the SIP URI in the To header contains the user's identity.

There are two parts to enabling this feature. First, you must enable the **register-use-to-for-lp** parameter in the global SIP configuration. Then you can set the next-hop in the applicable local policy attributes set to **ENUM**.

To enable your global SIP configuration for routing using the TO header:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **register-use-to-for-lp**—Set this parameter to enabled if you want the Oracle USM to use, for routing purposes, an ENUM query to return the SIP URI of the Registrar for a SIP REGISTER message. This parameters defaults to **disabled**.

   To set up your local policy attributes for routing using the TO header:

5. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

6. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

7. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

```
ORACLE(session-router)# local-policy
ORACLE(local-policy)#
```

8. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

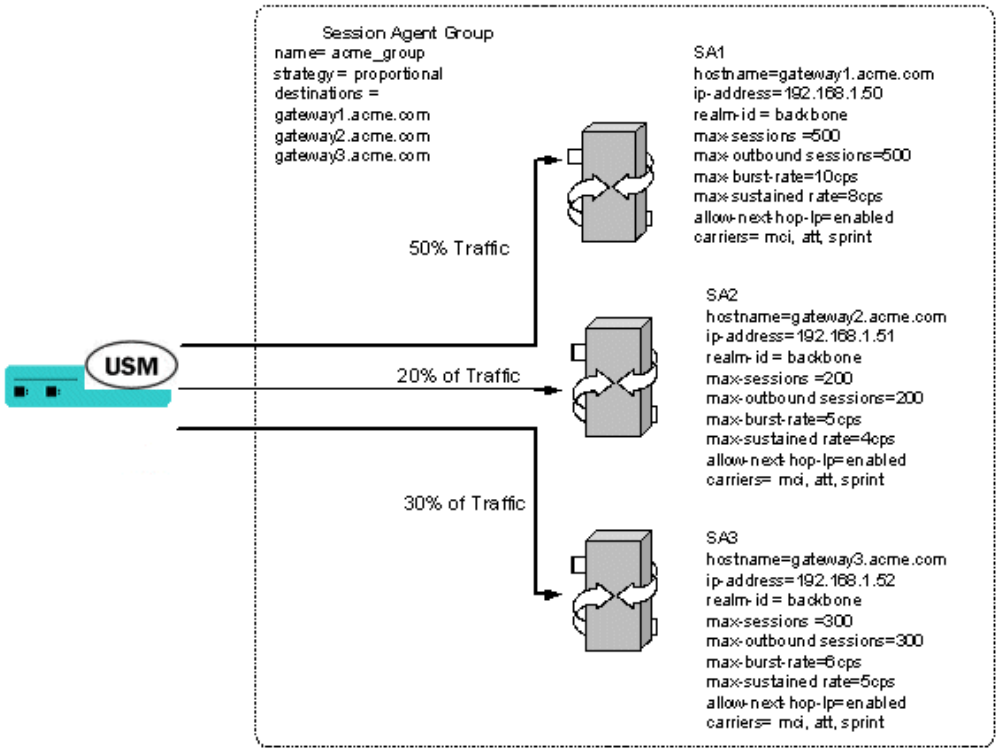
```
ORACLE(local-policy))# policy-attributes
ORACLE(policy-attributes)#
```

9. **next-hop**—This is the next signaling host. Set this parameter to ENUM if you want to use SIP methods in local policy attribute information for routing purposes.
10. Save and activate your configuration.

Load Balancing

This section describes Oracle USM load balancing. You can use session agent groups to assist in load balancing among session agents. You define concurrent session capacity and rate attributes for each session agent and then define the session agent group. Next, you select the allocation strategy you want applied to achieve the load balancing you want.

The following example shows a configuration for load balancing gateways based on a proportional allocation strategy.



Routing and load balancing capabilities include the following:

- least cost, which includes cost-based and time-based routing
- customer preference
- traffic aggregation
- routing by media (codec) type
- capacity-based, by destination
- service element load balancing
- service element failure detection and re-route
- session agent failure
- routing by codec

Configuring Routing

This section explains how to configure routing on the Oracle USM.

Configuration Prerequisite

You should have already configured the realms for your environment before you configure the routing elements. You need to know the realm identifier when configuring session agents and local policy.

You can use an asterisk (*) when the session agent exists in multiple realms.

Configuration Order

Recommended order of configuration:

- realm
- session agent
- session agent group
- local policy

Routing Configuration

You can enable, then configure, individual constraints that are applied to the sessions sent to the session agent. These constraints can be used to regulate session activity with the session agent. In general, session control constraints are used for session agent groups or SIP proxies outside or at the edge of a network. Some individual constraints, such as maximum sessions and maximum outbound sessions are not applicable to core proxies because they are transaction stateful, instead of session stateful. Other constraints, such as maximum burst rate, burst rate window, maximum sustained rate, and sustained rate are applicable to core routing proxies.

Configuring Session Agents

To configure session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type session-agent and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **host-name**—Enter the name of the host associated with the session agent in either hostname or FQDN format, or as an IP address.

   If you enter the host name as an IP address, you do not have to enter an IP address in the optional IP address parameter. If you enter the host name in FQDN format, and you want to specify an IP address, enter it in the optional IP address parameter. Otherwise you can leave the IP address parameter blank to allow a DNS query to resolve the host name.

   If the initial DNS query for the session agent fails to get back any addresses, the session agent is put out-of-service. When session agent is pinged, the DNS query is repeated. The ping message is not sent until the DNS query gets back one or more IP addresses. After the query receives some addresses, the ping message is sent. The session agent remains out of service until one of the addresses responds.

   > **Note:**
   >
   > The value you enter here must be unique to this session agent. No two session agents can have the same hostname.

   The hostnames established in the session agent populate the corresponding fields in other elements.

5. **ip-address**—Optional. Enter the IP address for the hostname you entered in FQDN format if you want to specify the IP address. Otherwise, you can leave this parameter blank to allow a DNS query to resolve the host name.

6. **port**—Enter the number of the port associated with this session agent. Available values include:

   - zero (0)—If you enter zero (0), the Oracle USM will not initiate communication with this session agent (although it will accept calls).
   - 1025 through 65535

     The default value is **5060**.

     > **Note:**
     >
     > If the transport method value is TCP, the Oracle USM will initiate communication on that port of the session agent.

7. **state**—Enable or disable the session agent by configuring the state. By default, the session agent is **enabled**.

   - enabled | disabled

8. **transport-method**—Indicate the IP protocol to use (transport method) to communicate with the session agent. **UDP** is the default value. The following protocols are supported:
   - **UDP**—Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts.
   - **UDP+TCP**—Allows an initial transport method of UDP, followed by a subsequent transport method of TCP if and when a failure or timeout occurs in response to a UDP INVITE. If this transport method is selected, INVITEs are always sent through UDP as long as a response is received.
   - **DynamicTCP**—dTCP indicates that dynamic TCP connections are the transport method for this session agent. A new connection must be established for each session originating from the session agent. This connection is torn down at the end of a session.
   - **StaticTCP**—sTCP indicates that static TCP connections are the transport method for this session agent. Once a connection is established, it remains and is not torn down.
   - **DynamicTLS**—dTLS indicates that Dynamic TLS connections are the transport method for this session agent. A new connection must be established for each session originating from the session agent. This connection is torn down at the end of a session.
   - **StaticTLS**—sTLS indicates that Static TLS connections are the transport method for this session agent. Once a connection is established, it will remain and not be torn down.
9. **realm-id**—Optional. Indicate the ID of the realm in which the session agent resides.

   The realm ID identifies the realm for sessions coming from or going to this session agent. For requests coming from this session agent, the realm ID identifies the ingress realm. For requests being sent to this session agent, the realm ID identifies the egress realm. In a Oracle USM, when the ingress and egress realms are different, the media flows must be steered between the realms.
   - no value: the egress realm is used unless the local policy dictates otherwise
   - asterisk (*): keep the egress realm based on the Request URI

   > **Note:**
   >
   > The realm ID you enter here must match the valid identifier value entered when you configured the realm.
10. **description**—Optional. Enter a descriptive name for this session agent.
11. **carriers**—Optional. Add the carriers list to restrict the set of carriers used for sessions originating from this session agent.

    Carrier names are arbitrary names that can represent specific service providers or traditional PSTN telephone service providers (for sessions delivered to gateways). They are global in scope, especially if they are exchanged in TRIP. Therefore, the definition of these carriers is beyond the scope of this documentation.

    You could create a list using carrier codes already defined in the North American Numbering Plan (NANP); or those defined by the local telephone number or carrier naming authority in another country.

> **Note:**
>
> If this list is empty, any carrier is allowed. If it is not empty, only local policies that reference one or more of the carriers in this list will be applied to requests coming from this session agent.

12. **allow-next-hop-lp**—Indicate whether this session agent can be used as a next hop in the local policy.

    If you retain the default value of **enabled**, the session agent can be used as the next hop for the local policy. Valid values are:

    - enabled | disabled

13. **constraints**—Enable this parameter to indicate that the individual constraints you configure in the next step are applied to the sessions sent to the session agent. Retain the default value of **disabled** if you do not want to apply the individual constraints. Valid values are:

    - enabled | disabled

    > **Note:**
    >
    > In general, session control constraints are used for SAGs or SIP proxies outside or at the edge of a network.

14. Enter values for the individual constraints you want applied to the sessions sent to this session agent. The following table lists the available constraints along with a brief description and available values.

Constraint	Description
maximum sessions	Maximum number of sessions (inbound and outbound) allowed by the session agent. The range of values is: • minimum: zero (0) is the default value and means there is no limit • maximum: 4294967295
maximum outbound sessions	Maximum number of simultaneous outbound sessions (outbound from the Oracle USM) that are allowed from the session agent. The range of values is: • minimum: zero (0) is the default value and means there is no limit • maximum: 4294967295 The value you enter here cannot be larger than the maximum sessions value.

Constraint	Description
maximum burst rate	Number of session invitations allowed to be sent to or received from the session agent within the configured burst rate window value. SIP session invitations arrive at and leave from the session agent in intermittent bursts. By entering a value in this field, you can limit the amount of session invitations that are allowed to arrive at and leave from the session-agent. For example, if you enter a value of 50 here and a value of 60 (seconds) for the burst rate window constraint, no more than 50 session invitations can arrive at or leave from the session agent in that 60 second time frame (window). Within that 60-second window, any sessions over the limit of 50 are rejected. The range of values is: • minimum: zero (0) session invitations per second • maximum: 4294967295 session invitations per second Zero is the is the default value.
maximum sustain rate	Maximum rate of session invitations (per second) allowed to be sent to or received from the session agent within the current window. The current rate is determined by counting the number of session invitations processed within a configured time period and dividing that number by the time period. By entering a value in this field, you can limit the amount of session invitations that are allowed to arrive at and leave from the session agent over a sustained period of time. For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes. For example, if you enter a value of 5000 here and a value of 3600 (seconds) for the sustain rate window constraint, no more than 5000 session invitations can arrive at or leave from the session agent in any given 3600 second time frame (window). Within that 3600-second window, sessions over the 5000 limit are rejected. The range of values is: • minimum: zero (0) invitations per second • maximum: 4294967295 invitations per second Zero is the is the default value. The value you set here must be larger than the value you set for the maximum burst rate constraint.
time to resume	Time in seconds after which the SIP proxy resumes sending session invitations to this session agent. This value only takes effect when the SIP proxy stops sending invitations because a constraint is exceeded. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero.

Constraint	Description
time to resume (ttr) no response	Delay in seconds that the SIP proxy must wait from the time that it sends an invitation to the session agent and gets no response before it tries again. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero. The value you enter here must be larger than the value you enter for the time to resume constraint.
in service period	Amount of time in seconds the session agent must be operational (once communication is re-established) before the session agent is declared as being in-service (ready to accept session invitations). This value gives the session agent adequate time to initialize. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295 seconds Default is zero.
burst rate window	Burst window period (in seconds) that is used to measure the burst rate. The term window refers to the period of time over which the burst rate is computed. Refer to the maximum burst rate information. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295seconds Zero is the is the default value. The value you set here must be smaller than the value you set for the maximum burst rate constraint.
sustain rate window	Sustained window period (in seconds) that is used to measure the sustained rate. Refer to the maximum sustain rate information. The range of values is: • minimum: zero (0) seconds • maximum: 4294967295seconds Zero is the is the default value. The value you set here must be larger than the value you set for the maximum sustain rate constraint.

15. **req-uri-carrier-mode**—SIP only. Set whether you want the selected carrier (determined by a value in the local policy) added to the outgoing message by configuring the request uri carrier mode parameter.

    You can set this parameter to let the system perform simple digit translation on calls sent to gateways. A 3-digit prefix is inserted in front of the telephone number (the Request-URI) that the gateway will use to select a trunk group. Most often, the Oracle USM needs to insert the carrier code into the signaling message that it sends on.

    The default value is **none**. The following lists the available modes.

    - **none**—Carrier information will not be added to the outgoing message.
    - **uri-param**—Adds a parameter to the Request-URI. For example, cic-XXX.
    - **prefix**—Adds the carrier code as a prefix to the telephone number in the Request-URI (in the same manner as PSTN).

16. **proxy-mode**—SIP only. Indicate the proxy mode to use when a SIP request arrives from this session agent.

    If this field is empty (upon initial runtime or upgrade), it's value is set to the value of the SIP configuration's proxy mode by default. If no proxy mode value was entered for the SIP configuration, the default for this field is **proxy**.

    The following are valid proxy modes:

    - **proxy**—If the Oracle USM is a Session Router, the system will proxy the request coming from the session agent and maintain the session and dialog state. If the Oracle USM is a Session Director, the system behaves as a B2BUA when forwarding the request.
    - **redirect**—The system sends a SIP 3xx reDIRECT response with contacts (found in the local policy) to the previous hop.

17. **redirect-action**—SIP only. Indicate the action you want the SIP proxy to take when it receives a Redirect (3XX) response from the session agent.

    The following table lists the available proxy actions along with a brief description

    - **recurse-305-only**—(default) If the Oracle USM receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All other 3xx responses are sent back to the previous hop.
    - **proxy**—The SIP proxy passes the response back to the previous hop; based on the proxy mode of the original request.
    - **recurse**—The SIP proxy serially sends the original request to the list of contacts in the Contact header of the response (in the order in which the contacts are listed in the response). For example, if the first one fails, the request will be send to the second, and so on until the request succeeds or the last contact in the Contact header has been tried.

18. **loose-routing**—SIP only. Enable this parameter if you want to use loose routing (as opposed to strict routing). The default is **enabled**. Valid values are:

    - enabled | disabled

      When the SIP NAT route home proxy parameter is enabled, the Oracle USM looks for a session agent that matches the home proxy address and checks the loose routing value. If loose routing is enabled, a Route header is included in the outgoing request in accordance with RFC 3261. If loose routing is disabled, the Route header is not included in the outgoing request (in accordance with strict routing procedures defined in RFC 2543).

      The loose routing value is also checked when the local policy's next hop value matches a session agent. If loose routing is set to enabled, the outgoing request retains the original Request-URI and Route header with the next hop address.

19. **send-media-session**—SIP only. Enable this parameter if you want to include a media session description (for example, SDP) in the INVITE or REINVITE message sent by the Oracle USM. Setting this field to **disabled** prevents the Oracle USM from establishing flows for that INVITE message.

    The default is **enabled**. Valid values are:

    - enabled | disabled

> **Note:**
>
> Only set send media session to disabled for a session agent that always redirects requests. It returns an error or 3xx response instead of forwarding an INVITE message.

20. **response-map**—Optional and for SIP only. Enter the name of the response map to use for this session agent. The mappings in each SIP response map is associated with a corresponding session agent. You can also configure this value for individual SIP interfaces.
21. **ping-method**—SIP only. Indicate the SIP message/method to use to ping a session agent. The ping confirms whether the session agent is in service. If this field is left empty, no session agent will be pinged.

    Setting this field value to the OPTIONS method might produce a lengthy response from certain session agents and could potentially cause performance degradation on your Oracle USM.
22. **ping-interval**—SIP only. Indicate how often you want to ping a session agent by configuring the ping interval parameter. Enter the number of seconds you want the Oracle USM to wait between pings to this session agent. The default value is **0**. The valid range is:

    - Minimum: 0
    - Maximum: 999999999

      The Oracle USM only sends the ping if no SIP transactions (have occurred to/from the session agent within the time period you enter here.
23. **trunk-group**—Enter up to 500 trunk groups to use with this single session agent. Because of the high number of trunk groups you can enter, the ACLI provides enhanced editing mechanisms for this parameter:

    - You use a plus sign (+) to add single or multiple trunk groups to the session agent's list.

      When you add a single trunk group, simply use the plus sign (+) in front of the trunk group name and context. Do not use a Space between the plus sign and the trunk group name and context.

      For example, you might have already configured a list of trunk groups with the following entries: **tgrpA:contextA, tgrpB:contextB,** and **tgrpC:contextC**. To add **tgrp1:context1**, you would make the following entry:

      ```
      ORACLE(session-agent)# trunk-group +tgrp1:context1
      ```

      Your list would then contain all four trunk groups.

      When you add multiple trunk groups, simply enclose your entry in quotation marks () or in parentheses (()). While you put spaces between the trunk group name and context entries, you do not use spaces with the plus sign, parentheses or quotation marks.

      ```
      ORACLE(session-agent)# trunk-group +tgrp1:context1 tgrp2:context2
      tgrp3:context3
      ```
    - You use a minus sign (-) to delete single or multiple trunk groups from the session agent's list.

      When you remove a single trunk group, simply use the minus sign (-) in front of the trunk group name and context. Do not use a Space between the minus sign and the trunk group name and context.

For example, you might have already configured a list of trunk groups with the following entries: tgrpA:contextA, tgrpB:contextB, tgrpC:contextC, and tgrp1:context1. To delete tgrp1:context1 from the list, you would make the following entry:

```
ORACLE(session-agent)# trunk-group -tgrp1:context1
```

Your list would then contain: tgrpA:contextA, tgrpB:contextB, and tgrpC:contextC.

When you add multiple trunk groups, simple enclose your entry in quotation marks () or in parentheses (()). While you put spaces between the trunk group name and context entries, you do not use spaces with the plus sign, parentheses or quotation marks.

```
ORACLE(session-agent)# trunk-group -tgrp1:context1 tgrp2:context2
```

- You overwrite (replace) the entire list of a session agent's trunk groups by entering a list that does not use either the plus (+) or the minus (-) sign syntax.

24. **ping-in-service-response-codes**—SIP only. Enter the list of response codes that keep a session agent in service when they appear in its response to the Oracle USM's ping request. The Oracle USM takes the session agent out of service should a response code be used that does not appear on this list. Default is **none**.
25. **out-service-response-codes**—SIP only. Enter the list defines the response codes that take a session agent out of service when they appear in its response to the Oracle USM's ping request or any in-dialog creating request (such as an INVITE, SUBSCRIBE, etc.). The Oracle USM ignores this list if an in-service list exists.
26. **options**—Optional. You can add your own features and/or parameters by using the options parameter. You enter a comma-separated list of either or both of the following:

    - feature=<value feature>

    For example:

    You can include the original address in the SIP message from the Oracle USM to the proxy in the Via header parameter by entering the following option:

    via-origin=<parameter-name>

    The original parameter is included in the Via of the requests sent to the session agent. The via origin feature can take a value that is the parameter name to include in the Via. If the value is not specified for via origin, the parameter name is origin.

    > **Note:**
    >
    > If the feature value itself is a comma-separated list, enclose it within quotation marks.

27. **in-translationid**—Optional. Enter the In Translation ID for a configured session translation (group of address translation rules with a single ID) if you want to apply session translation to incoming traffic.
28. **out-translationid**—Optional. Enter the Out Translation ID for a configured session translation (group of address translation rules with a single ID) if you want to apply session translation to outgoing traffic.

    Address translations attached to session agents take precedence over address translations attached to realms. If no address translation is applied to a session agent, then the Oracle USM will use the address translation applied to a realm. If an address translation is applied to both a realm and session agent, the translation attached to the session agent will apply. If

the applicable session agent and realm have no associated translations, then the addresses will remain in their original forms and no address translations will be performed.

29. **trust-me**—Indicate whether this session agent is a trusted source, which the Oracle USM checks when it receives a message to determine if the source is trusted. The default value is **disabled**. The valid values are:

    - enabled | disabled

    The following example shows a session agent with an IP address used for the hostname.

```
session-agent
        hostname                        192.168.1.10
        ip-address                      192.168.1.10
        port                            5060
        state                           enabled
        app-protocol                    SIP
        app-type
        transport-method                UDP
        realm-id                        realm-1
        description                     englab
        carriers
                                        carrier1
        allow-next-hop-lp               enabled
        constraints                     disabled
        max-sessions                    355
max-inbound-sessions            4
        max-outbound-sessions           355
        max-burst-rate                  0
        max-inbound-burst-rate          10
        max-outbound-burst-rate         1
        max-sustain-rate                3000
        max-inbound-sustain-rate        0
        max-outbound-sustain-rate       0
        min-seizures                    5
        min-asr                         0 time-to-resume                  60
        ttr-no-response                 0
        in-service-period               30
        burst-rate-window               60
        sustain-rate-window             3600
        req-uri-carrier-mode            None
        proxy-mode                      Proxy
        redirect-action                 Recurse
        loose-routing                   enabled
        send-media-session              enabled
        response-map
        ping-method
        ping-interval                   0
        media-profiles
        in-translationid
        out-translationid
        trust-me                        disabled
        request-uri-headers
        stop-recurse
        local-response-map
        ping-to-user-part
        ping-from-user-part
        li-trust-me                     disabled
        in-manipulationid
        out-manipulationid
        p-asserted-id
```

```
trunk-group
max-register-sustain-rate      0
```

Session Agent Groups Configuration

To configure session agent groups:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-group** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters. This value cannot contain underscores.

   ```
   ORACLE(session-router)# session-group
   ORACLE(session-agent-group)#
   ```

4. **group-name**—Enter a unique name for the session agent group in Name format.
5. **description**—Optional. Enter descriptive information about the session agent group.
6. **state**—Enable or disable the session agent group on the Oracle USM. The default value is **enabled**. Valid values are:
   - enabled | disabled
7. **application-protocol**—Indicate the signaling protocol you want to use with the session agent group. The default value is **SIP**. The valid values are:
   - SIP
8. **strategy**—Indicate the session agent allocation strategy you want to use. The strategy you chose selects the session agents that will be made available by this session agent group. The default value is **hunt**. The valid values are:
   - **hunt**—Selects session agents in the order in which they are listed. For example, if the first agent is online, working, and has not exceeded defined constraints, all traffic is sent to the first agent. If the first agent is offline or if it exceeds a defined constraint, the second agent is selected. If the first and second agents are offline or exceed defined constraints, the third agent is selected. And so on through the list of session agents.
   - **roundrobin**—Selects each session agent in the order in which they are listed in the destination list, selecting each agent in turn, one per session.
   - **leastbusy**—Selects the session agent that has the fewest number of sessions relative to the maximum sessions constraint (for example, lowest percent busy) of the session agent element. The Least Busy Calculation is the result of dividing the number of active calls for a session agent by the max-sessions parameter within the session-agent element configuration. If the default max-session parameter value issued for a session agent (0), the result of the Least Busy Calculation will be 0. The Least Busy SAG Strategy will route a session to the session agent with the lowest resulting Least Busy Calculation percentage. If multiple session agents have the lowest percentage, the foremost session agent in the Session Agent Group dest parameter will be used.
   - **propdist**—Based on programmed, constrained session limits, the Proportional Distribution strategy proportionally distributes the traffic among all of the available session agents. Sessions are distributed among session agents based on the max-outbound-sessions value in each session agent. The sum of max-outbound-sessions for

every session-agent within a session group equates to 100% and the max-outbound-sessions value for each session-agent represents a % that total. Sessions are proportionally allocated to session agents based on their individual session agent max-outbound-sessions value, as a % of the total max-outbound-sessions for the group.

- **lowsusrate**—The Lowest Sustained Rate strategy routes to the session agent with the lowest sustained rate of session initiations/invitations (based on observed sustained session request rate).

9. **destination**—Identify the destinations (session agents) available for use by this session agent group.

   A value you enter here must be a valid IP address or hostname for a configured session agent.

10. **trunk-group—**Enter trunk group names and trunk group contexts to match in either IPTEL or custom format. If left blank, the Oracle USM uses the trunk group in the realm for this session agent group. Multiple entries are surrounded in parentheses and separated from each other with spaces.

    Entries for this list must one of the following formats: trgp:context or trgp.context.

11. **sag-recursion**—Enable this parameter if you want to use SIP SAG recursion for this SAG. The default value is **disabled**. Valid values are:

    - enabled | disabled

12. **stop-sag-recurse**—Enter the list of SIP response codes that terminate recursion within the SAG. Upon receiving one of the specified response codes, such as 401 unauthorized, or upon generating one of the specified response codes internally, such as 408 timeout, the Oracle USM returns a final response to the UAC. You can enter the response codes as a comma-separated list or as response code ranges.

    The following example shows a session agent group using the SIP protocol and with the Hunt allocation strategy applied.

```
session-group
group-name                     proxy-sag1
description                    proxies for external domain
state                          enabled
app-protocol                   SIP
strategy                       Hunt
dest                           gw1
                               gw2

trunk-group
tgname2:tgcontext2
sag-recursion                  disabled
stop-sag-recurse               401,407
last-modified-date             2005-01-09 23:23:36
```

SAG Matching for LRT and ENUM

When this feature is enabled and a match is found, the Oracle USM uses the matching SAG for routing. When there is no match for the SAG, the Oracle USM processes the result as it would have if this feature had not been enabled: either matching to a session agent hostname, or performing a DNS query to resolve it.

Note that you set the state of this feature in the SIP configuration.

To configure a SAG for ENUM or LRT matching:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) that configuration to edit it.

4. **enum-sag-match**—Set this parameter to enabled so the Oracle USM will match session agent group (SAG) names with the hostname portion in the naming authority pointer (NAPTR) from an ENUM query or LRT next-hop entry. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Configuring Local Policy

To configure local policy:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ACMEPACKET(session-router)# local-policy
ACMEPACKET(local-policy)#
```

4. **from-address**—Indicate the originating address information by entering a From address value. You can use the asterisk (*) as a wildcard to indicate this policy can be used with all originating addresses.

   You can also use complete or partial E.164 addresses (strings that contain telephone keypad characters) here. Number matching works from left to right. Formats include the following:
   - SIP From address
   - FQDNs
   - IP addresses
   - 123*456

     The Oracle USM also supports the asterisk as part of the From address you configure in your local policies.

     This means that for the **from-address** parameters of a local policy configuration, you can enter values in which an asterisk appears and match them accordingly. You might enter a value that resemble the following example:

> **Note:**
>
> After entering the from-address value, the Oracle USM automatically saves it to the configuration when exiting from localpolicy.

5. **to-address**—Indicate the destination address by entering a To address value. You can use the asterisk (*) as a wildcard to indicate all this policy can be used for any destination address.

   You can also use E.164 addresses (strings that contain telephone keypad characters) here. Number matching works from left to right. Formats include the following:

   - SIP Request-URI
   - FQDNs
   - IP addresses
   - 123*456

   The Oracle USM also supports the asterisk as part of the To address you configure in your local policies.

   This means that for the **to-address** parameters of a local policy configuration, you can enter values in which an asterisk appears and match them accordingly. You might enter a value that resembles the following example:

> **Note:**
>
> After entering the to-address value, the Oracle USM automatically saves it to the configuration when exiting from localpolicy.

6. **source-realm**—Enter the realm, or list of realms, you want the Oracle USM to use to determine how to route traffic. This list identifies from what realm traffic is coming and is used for routing by ingress realm by the local policy.

   You can use the asterisk (*) as a wildcard to indicate this local policy can be used with all realms. The default value is *.Or you can enter a value that corresponds to the identifier of an already configured realm. Formats include the following:

   - realm ID
   - customer name
   - peer name
   - subdomain name
   - VPN identifier

7. **activate-time**—Set the time you want the local policy to be activated using the following syntax:

   ```
   yyyy:mm:dd hh:mm:ss
   yyyy:mm:dd-hh:mm:ss
   ```

8. **deactivate-time**—Set the time you want the local policy to be deactivated using the following syntax:

   ```
   yyyy:mm:dd hh:mm:ss
   yyyy:mm:dd-hh:mm:ss
   ```

9. **state**—Indicate whether you want the local policy to be enabled or disabled on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
10. **policy-attribute**—Configure local policy attributes by following steps 8 through 21.
11. **next-hop**—Identify the next signaling host by entering the next hop value. You can use the following as next hops:
    - IPv4 address or IPv6 address of a specific endpoint
    - Hostname or IPv4 address or IPv6 address of a configured session agent
    - Group name of a configured session agent group

    > **Note:**
    >
    > The group name of a configured session agent group must be prefixed with SAG:

    For example:

    policy-attribute

    next-hop SAG:appserver

    policy-attribute

    next-hop lrt:routetable

    policy-attribute

    next-hop enum:lerg

    You can also configure a next hop that has an address of 0.0.0.0, thereby creating a null route. Different from not having a local policy configured (which would trigger Oracle USM local policy recursion), this terminates local policy recursion and immediately fails the request. In these cases, the Oracle USM responds a request with a 404 Not Found.
12. **realm**—Identify the egress realm (the realm used to reach the next hop) if the Oracle USM must send requests out from a specific realm.

    The value you enter here must correspond to a valid identifier you enter when you configured the realm. If you do not enter a value here, and the next hop is a session agent, the realm identified in the session agent configuration is used for egress.
13. **replace-uri**—Indicate whether you want to replace the Request-URI in outgoing SIP requests with the next hop value.
14. **carrier**—Optional. Enter the name of the carrier associated with this route. The value you enter here must match one or more of the carrier names in the session agent configuration.

    Entries in carrier fields can be from 1 to 24 characters in length and can consist of any alphabetical character (Aa-Zz), numerical character (0-9), or punctuation mark (! ” # $ % ^ & * ( ) + - = < > ? ‘ | { } [ ] @ / \ ‘ ~ , . _ : ;) or any combination of alphabetical characters, numerical characters, or punctuation marks. For example, both 1-0288 and acme_carrier are valid carrier field formats.
15. **start-time**—Indicate the time of day (from the exact minute specified) the local policy attributes go into effect. Enter only numerical characters (0-9) and follow the 4-digit military time format. For example:

```
1400
```

The default value of **0000** implies that the defined policy attributes can be considered in effect any time after 00:00:00. The valid range is:

- Minimum—0000
- Maximum—2400

16. **end-time**—Indicate the time of day (from the exact minute specified) the local policy attributes are no longer in effect. Enter only numerical characters (0-9) and follow the 4-digit military time format. For example:

    ```
    2400
    ```

    The default value of **2400** implies that the defined policy attributes can be considered in effect any time before midnight. The valid range is:

    - Minimum—0000
    - Maximum—2400

17. **days-of-week**—Enter any combination of days of the week (plus holidays) you want the local policy attributes to be in effect. You must enter at least one day or holiday here. A holiday entry must correspond with a configured holiday established in the Session Router.

    The default is **U-S**. The valid values are:

    - U (Sunday)
    - M (Monday)
    - T (Tuesday(
    - W (Wednesday)
    - R (Thursday)
    - F (Friday)
    - S (Saturday)
    - H (Holiday)

      You can enter a range of values separated by a hyphen, for example U-S. And you can enter multiple values separated by commas, for example M,W,F. You cannot use spaces as separators.

18. **cost**—Enter a cost value that acts as a unitless representation of the cost of a route relative to other routes reaching the same destination (To address). This value is used as a way of ranking policy attributes.

    The default value is zero (**0**). The valid values are:

    - minimum—zero (0)
    - maximum—999999999

19. **state**—Indicate whether you want to enable or disable the local policy. The default value is **enabled**. The valid values are:

    - enabled | disabled

20. **media-profiles**—Configure a list of media profiles if you want the local policy to route SIP traffic by the codecs specified in the SDP. The list of media profiles entered here are matched against the SDP included in SIP requests and the next hop is selected by codec.

The values in this list are matched against the rtpmap attribute of passed SDP, and preference weight for route selection is based on the order in which the matching payload type appears in the SDP's media (m=) line.

For example when the following SDP arrives:

```
m=audio 1234 RTP/AVP 0 8 18
```

that contains the following attributes that correspond to three configured local policies with the same cost:

- a=rtpmap:**0** PCMU/8000
- a=rtpmap:**8** PCMA/8000
- a=rtpmap:**18** G729/8000

the following route selection action occurs:

The local policy route that corresponds to the a=rtpmap:**0** PCMU/8000 attribute is selected because the payload type of **0** in the attribute line matches the first payload type of 0 listed in the m= line. The codec value of PCMU indicated in this selected attribute is used to find the local policy with the media profiles attribute that includes PCMU in the list.

Because the value you enter here is matched against the codec values included in the actual passed SDP, it must correspond to accepted industry-standard codec values.

The following example shows a local policy with a next hop value of the session agent group called gw-sag2.

```
local-policy
        from-address                            *
        to-address
                                                192.168.1.10
        source-realm                            *
        activate-time                           2005-01-20 20:30:00
        deactivate-time                         N/A
        state                                   enabled
        last-modified-date                      2005-01-10 00:36:29
policy-attribute
        next-hop                                SAG:gw-sag2
        realm
        replace-uri                             enabled
        carrier
        start-time                              0000
        end-time                                2400
        days-of-week                            U-S
        cost                                    0
        app-protocol
        state                                   enabled
        media-profiles
```

Local Policy Matching for Parent Realms

You can configure the Oracle USM to use the parent realm for routing purposes even when the source realm for an incoming message is a child realm.

With this feature disabled (default), the Oracle USM uses the specific source realm to perform a local policy look-up. When the source realm is a child realm and any relevant local policies are configured with the parent realm, there will be no matches and the local policy look-up will

fail. To avoid this issue and ensure successful look-ups, you must configure multiple local policies if you want to use a configuration with nested realms.

The Oracle USM examines the source realm to determine if it is a parent realm with any child realms when you enable this feature. If the parent, source realm does have child realms, then the Oracle USM creates local policy entries for the parent and all of its child realms. This operation is transparent and can save time during the configuration process.

It is possible, then, for a local policy look-up to match the same child realm in two ways:

- Through a match via the parent realm
- Through a direct match for a local policy configured with that specific child realm

In such a case, the child realm must have different costs for each type of match to avoid collisions.

This feature is enabled on a global basis in the session router configuration. Because it applies system-wide, all source realms will use this form of matching when enabled.

To enable local policy matching for parent realms:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-router** and press Enter.

   ```
   ORACLE(session-router)# session-router
   ORACLE(session-router-config)#
   ```

4. **match-lp-source-parent-realms**—If you want the Oracle USM to perform local policy realm matching based on the parent realm (so that there are local policy entries for parent and child realms), set this parameter to **enabled**. The default value is **disabled**. The valid values are:

   - enabled | disabled

   ```
   ORACLE(session-router-config)# match-lp-src-parent-realms enabled
   ```

5. Save and activate your configuration.

SIP Session Agent DNS-SRV Load Balancing

Prior to Release 6.2.0 the Oracle USM provided the ability to specify an FQDN (fully qualified domain name) for a destination session-agent. During DNS lookup the FQDN could resolve to multiple SRV (Resource Record for Servers) records. Each SRV could resolve to a single IP address via A-Record query in IMS or DNS.

With Release 6.2.0 the Oracle USM supports load balancing behavior as described in RFC 3263, Session Initiation Protocol (SIP): Locating SIP Servers.

The Oracle USM will provide a new parameter **ping-all-addresses** in session-agent configuration mode to enable internal load balancing and RFC 3263 compliance. The Oracle USM monitor the availability of the dynamically resolved IP addresses obtained from DNS server using OPTIONS ping (ping-per-DNS entry). The ping-method and ping-interval for each resolved IP addresses will be copied from original session-agent.

Status of Session-Agent:

In Service – if any of dynamically resolved IP addresses is in service

Out of service – if all dynamically resolved IP addresses is out of service.

The default of **ping-all-addresses** is disabled, in which case the Oracle USM only pings the first available resolved IP addresses.

With status of each resolved IP addresses above, the Oracle USM will recurse through the list of these in-service IP addresses dynamically resolved from DNS server on 503 response, and stop recursion based upon a configured list of response values specified by the **stop-recurse** parameter in sip-interface configuration mode. With internal load balancing enabled in the session-agent, the Oracle USM provides the ability to select routing destinations based on SRV weights. The priority/weight algorithm is based on RFC 2782, *A DNS RR for specifying the location of services (DNS SRV)*.

The Oracle USM will provide the similar functionality as that listed above for A-records, the Oracle USMwill select first available routing destinations because there is no priority/weight contained in A-records.

Session Agent DNS-SRV Load Balancing Configuration

To configure the Oracle USM to perform Session-Agent DNS-SRV load balancing:

1. From superuser mode, use the following command sequence to access sip-config configuration mode. While in this mode, you configure SAG-based address resolution.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

2. Use the **ping-all-addresses** parameter to enable Session-Agent DNS-SRV load balancing.
3. Use **done**, **exit**, and **verify-config** to complete Session-Agent DNS-SRV load balancing configuration.

   The **show sip agents** ACLI command displays the availability of dynamically resolved IP addresses

   ```
   ORACLE# show sip agents acme.engr.com
   21:46:05-51-router
   Session Agent acme.engr.com(core) [In Service] NO ACTIVITY

   ... Statistics of ALL IPs associated with this SA ....

   Destination: 192.168.200.235 In Service
   Destination: 192.168.200.231 In Service
   ```

Configurable DNS Response Size

When a realm is used for DNS queries, the Oracle USM can accept UDP DNS responses configurable up to 65535 bytes.

This functionality is useful when large numbers of SRV records will be returned in a DNS query thereby eliciting a large-sized DNS response. This behavior should be configured on the realm where the DNS servers are located.

To extend the valid DNS response size, add the **dns-max-response-size** option to the realm configuration. If this option is not configured, the Oracle USM uses the default maximum response size of 512 bytes, and information past the 512th byte will be ignored.

Do not add the **dns-max-response-size** option to realms where DNS queries are not being performed. Ensure that the realm where this option is configured is referenced in a transport realm's **dns-realm** parameter. Only the local value of the **dns-max-response-size** option is used for the realm; there is no inheritance of this value.

DNS Response Size Configuration

1. Access the **realm-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

2. Select the **realm-config** object to edit.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: realm01 left-left:0 0.0.0.0

   selection: 1
   ORACLE(realm-config)#
   ```

3. Add the **dns-max-response-size** option to the realm with a value between 513 — 65535.

   ```
   ORACLE(realm-config)#options dns-max-response-size=4196
   ```

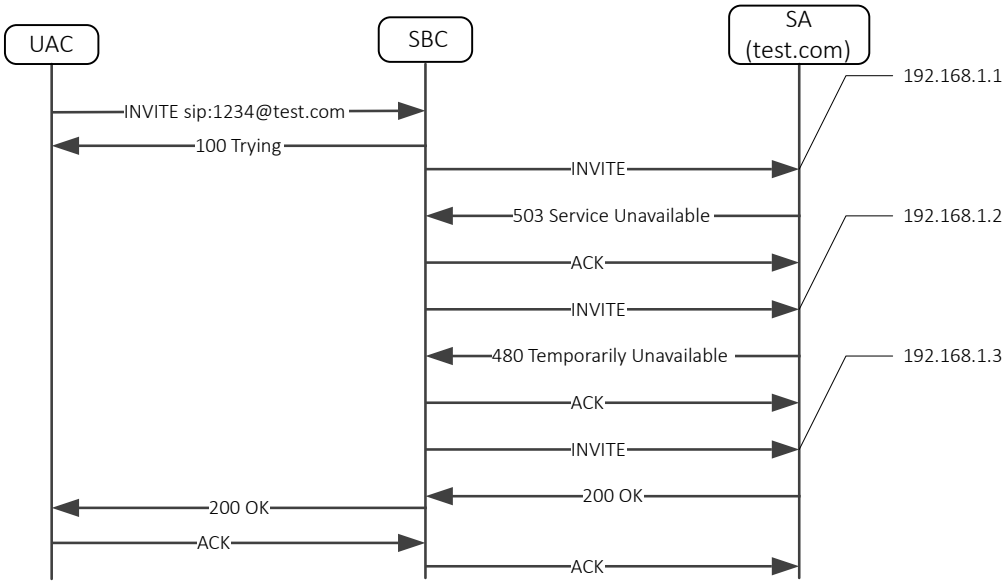
4. Type **done** to save your configuration.

DNS-SRV Session Agent Recursion Error Handling

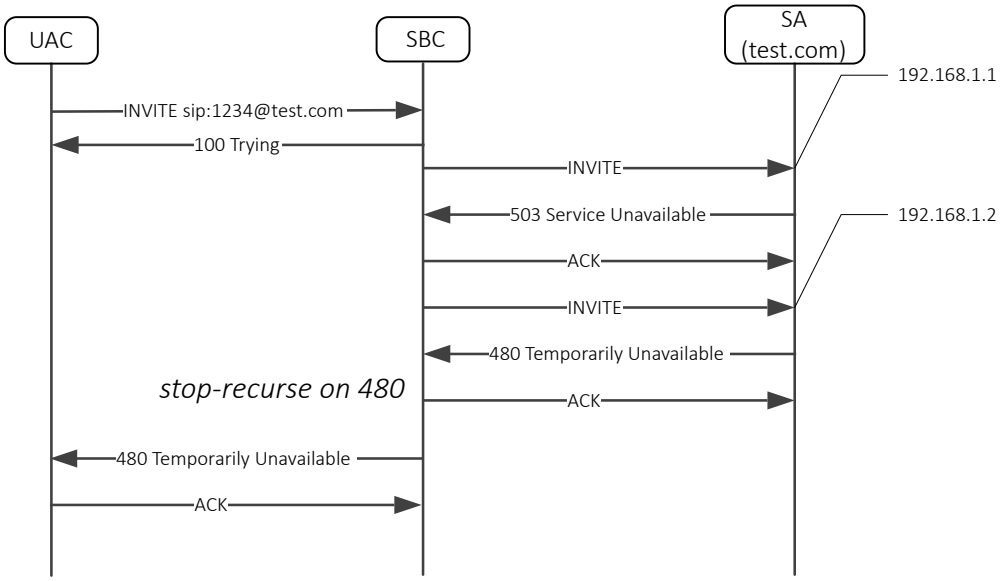
When a session request is sent from the Oracle USM to a session agent, and an error response is received (or a transport failure occurs), the Oracle USM attempts to reroute the message through the list of dynamically resolved IP addresses. The SBC can be configured to resend session requests through the list of IP addresses under more failure conditions.

This feature concerns the case when a session agent is configured with an FQDN in the hostname parameter and the **dns-load-balance** or **ping-all-addresses** option is configured. This configuration sets up the load balancing / redundancy behavior for the SBC to use all addresses returned in the SRV/A-record for that session agent. In previous versions of the SBC software, only when a 503 failure from the SA was received would the SBC resend the session request to the next dynamically resolved IP address (on the SRV/A record list).

By adding the **recurse-on-all-failures** option to a session agent, the Oracle USM will resend a session request to the next address on the list after a 4xx or 5xx failure response has been received from a session agent.



If the SBC receives a failure response from the session agent, and the number of that failure is configured in the **stop-recurse** parameter, no further session requests will be forwarded to additional addresses from the SRV/A record list. The error message will be forwarded back to the UA.



Answer to Seizure Ratio-Based Routing

New SIP session agent constraints set a threshold for Answer to Seizure Ratio (ASR) has been implemented. ASR is considered when determining whether session agents are within their constraints to route calls (in addition to session and rate constraints).

The new session agent constraints indicate the minimum acceptable ASR value and computes the ASR while making routing decisions. ASR is calculated by taking the number of successfully answered calls and dividing by the total number of calls attempted (which are known as seizures).

If the ASR constraints are exceeded, the session agent goes out of service for a configurable period of time and all traffic is routed to a secondary route defined in the local policy (next hop with higher cost).

The two session agent constraints are:

- minimum seizure: determines if the session agent is within its constraints. When the first call is made to the session agent or the if calls to the session agent are not answered, the minimum seizure value is checked.
  For example, if 5 seizures have been made to the session agent and none of them have been answered, the sixth time, the session agent is marked as having exceeded its constraints and the calls will not be routed to it until the time-to-resume has elapsed.
- minimum ASR: considered when make routing decisions. If some or all of the calls to the session agent have been answered, the minimum ASR value is considered to make the routing decisions.
  For example, if the you set the minimum ASR at 50% and the session agent's ASR for the current window falls below 50%, the session agent is marked as having exceeded its constraints and calls will not be routed to it until the time-to-resume has elapsed.

ASR Constraints Configuration

To configure ASR constraints:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. If configuring an existing session agent, enter the select command to select the session agent.
5. **min-seizures**—Enter the minimum number of seizures that when exceeded, cause the session agent to be marked as having exceeded its constraints. Calls will not be routed to the session agent until the time-to-resume has elapsed. The default value is **5**. The valid range is:
   - Minimum—1
   - Maximum—999999999
6. **min-asr**—Enter the percentage you want as the minimum. If the session agent's ASR for the current window falls below this percentage, the session agent is marked as having exceeded its constraints and calls will not be routed to it until the time-to-resume has elapsed. The default value is **0**. The valid range is:
   - Minimum—0
   - Maximum—100
7. Save and activate your configuration.

   The following example shows a session agent configuration.

```
session-agent
        hostname                        192.168.1.6
        ip-address
        port                            1720
        state                           enabled
        app-protocol                    H323
        app-type                        H323-GW
        transport-method
        realm-id                        external
        description
        carriers
        allow-next-hop-lp               enabled
        constraints                     enabled
        max-sessions                    0
max-inbound-sessions            4
        max-outbound-sessions           5
        max-burst-rate                  0
        max-inbound-burst-rate          10
        max-outbound-burst-rate         1
        max-sustain-rate                0
        max-inbound-sustain-rate        0
        max-outbound-sustain-rate       0
min-seizures                    5
        min-asr                         50
        time-to-resume                  30
        ttr-no-response                 0
        in-service-period               0
        burst-rate-window               0
        sustain-rate-window             0
        req-uri-carrier-mode            None
        proxy-mode
        redirect-action
        loose-routing                   enabled
        send-media-session              enabled
        response-map
        ping-method
        ping-interval                   0
        media-profiles
        in-translationid
        out-translationid
        trust-me                        disabled
        request-uri-headers
        stop-recurse
        local-response-map
        ping-to-user-part
        ping-from-user-part
        li-trust-me                     disabled
        in-manipulationid
        out-manipulationid
        p-asserted-id
        trunk-group
        max-register-sustain-rate       0
        early-media-allow
        invalidate-registrations        disabled
        last-modified-date              2006-05-12 19:48:06
```

SIP Recursion Policy

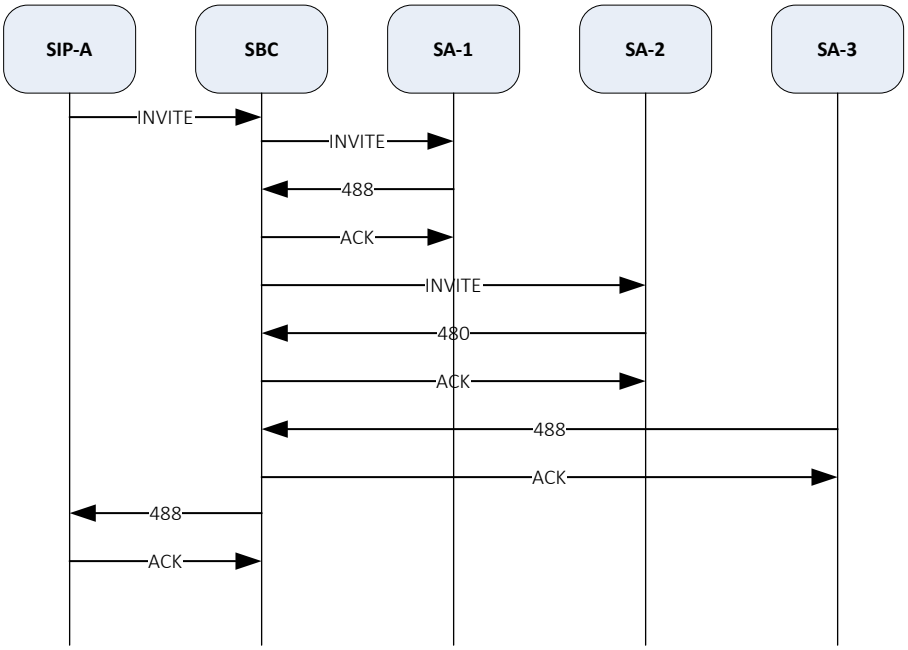
Session Agents (and Session agent groups) can utilize a SIP Recursion policy to customize the Oracle USM behavior when recursing through a list of target SIP peers. These policies are

useful for networks with large numbers of SIP peers that wish to customize recursive routing behavior for individual session agents or session agent groups, based on number of recursion attempts or the returned SIP response code.

SIP recursion policy provides the Oracle USM with rules to indicate when to resend messages to the next SIP peer based on the previous response before terminating recursion and forwarding the final peer's response back to the initial requester.

The system terminates recursing among session agents (or session agent groups) when one of two criteria is first met. The first criteria is reaching a maximum number of recursions among all targets for a given SIP request. The second criteria is receiving a maximum number of specific response codes from all targets as recursed for a given SIP request.

To set a maximum number of recursions before forwarding the final response back to the ingress-side requester, you set the **sip-recursion-policy** > **global-count** parameter. Once the Oracle USM exceeds retrying to send a message to valid targets this number of times, it stops recursing and forwards the final response back to the requester. To disable maximum recursion hops per-call, set the **sip-recursion-policy** > **global-count** to 0.



SIP recursion policy can also terminate recursion based on identifying one or more specific 3xx, 4xx, or 5xx response messages received from all targets in response to the request. The Oracle USM can consider the number of responses per response code that were received before terminating recursion.

In absolute mode, the Oracle USM stops recursing after the total configured number of responses (**sip-response-code** > **attempts**) for a configured **sip-response-code** subelement has been received. When the number of attempts per response code is received, recursion stops and the final response is forwarded to the requestor. Considering the previous call flow, in the following configuration example, after the Oracle USM receives the 2nd 488 response from among all targets, it terminates recursion and forwards the last 488 to the requester:

```
sip-response-code
                response-code                   488
                attempts                        2
```

Consecutive mode sets the Oracle USM to stop recursing after the total configured number of responses (**sip-response-code** > **attempts**) for a configured **sip-response-code** subelement has been received consecutively. When the number of attempts per response code is received, recursion stops and the final response is forwarded to the requestor. Considering the previous example's configuration set to consecutive mode, the Oracle USM will need to receive two 488 responses in a row before recursion terminates and the last 488 is forwarded back to the requester.

Final Configuration

In order to use a **sip-recursion-policy**, it must be configured within a session agent or session agent group. Populate a **session-agent-group** > **sip-recursion-policy** or **session-agent** > **sip-recursion-policy** parameter with the **name** of the SIP recursion policy you wish to apply.

Feature Interactions

The existing SIP Configurable Route Recursion feature takes higher precedence than this feature. If the **stop-recurse** parameter is configured in the SIP interface, session agent, or SAG, the Oracle USM checks each response against that list, before any SIP recursion policy.

SIP Recursion Policy Configuration

1. Access the **sip-recursion-policy** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-recursion-policy
   ORACLE(sip-recursion-policy)#
   ```

2. name - Set a name for this SIP Recursion Policy. This value will be referenced by individual session agents' or session agent groups' **sip-recursion-policy** parameter.
3. description - Enter a textual description of this SIP Recursion Policy instance. If the description includes spaces, enclose all words within double quotes.
4. global-count - Enter the maximum number of recursions to take before terminating recursion and sending the response back to the requester. Entering 0 here disables a maximum recursion counter.
5. mode - If terminating recursion based on responses received, enter whether to terminate on absolute responses or consecutive responses. This value is set if also configuring **sip-response-code** subelements. Valid values are:
   - absolute
   - consecutive
6. Type **done** to save your configuration.

If configuring the Oracle USM to terminate recursion on received response, continue this procedure to create one or more **sip-response-code** subelement.

7. Access the **sip-response-code** subelement.

   ```
   ORACLE(sip-recursion-policy)# sip-resp-code-attempts
   ORACLE(sip-response-code)#
   ```

8. response-code - Enter the SIP response code number to associate with an attempt number through this configuration element for when to terminate recursion.
   - Default: 503
   - Range: 300 - 599

9. attempts - Enter the number of times the system must receive a message of the **response-code** before terminating recursion. This value is handled according to the **sip-recursion-policy** > **mode** parameter.
   - Default: 1
   - Range: 1 - 1000
10. Type **done** to save your configuration.

Now, configure a the appropriate **session-agent** > **sip-recursion-policy** or **session-agent-group** > **sip-recursion-policy** parameter with the **name** of the SIP Recursion policy you just created.

ENUM Lookup

Telephone Number Mapping (ENUM from TElephone NUmber Mapping) is a suite of protocols used to unify the telephone system with the Internet by using E.164 addresses with the Domain Name System (DNS). With ENUM, an E.164 number can be expressed as a Fully Qualified Domain Name (FQDN) in a specific Internet infrastructure domain defined for this purpose (e164.arpa). E.164 numbers are globally unique, language independent identifiers for resources on Public Switched Telecommunication Networks (PSTNs). ITU-T recommendation E.164 is the international public telecommunication telephony numbering plan.

How ENUM Works

ENUM uses DNS-based architecture and protocols for mapping a complete international telephone number (for example, +1 202 123 1234) to a series of Uniform Resource Identifiers (URIs).

The protocol itself is defined in the document E.164 number and DNS (RFC 3761) that provides facilities to resolve E.164 telephone numbers into other resources or services on the Internet. The syntax of Uniform Resource Identifiers (URIs) is defined in RFC 2396. ENUM uses Naming Authority Pointers (NAPTR) records defined in RFC 2915 in order to identify available ways or services for contacting a specific node identified through the E.164 number.

Translating the Telephone Number

A telephone number is translated into an Internet address using the following steps:

1. The number is first stored in the following format, +1-202-555-1234. 1 is the country code for the United States, Canada, and the seventeen other countries that make up the North American Numbering Plan (NANP). The + indicates that the number is a complete, international E.164 telephone number.
2. All characters are removed except for the digits. For example, 12025551234.
3. The order of the digits is reversed. For example, 43215552021. The telephone number is reversed because DNS reads addresses from right to left, from the most significant to the least significant character. Dots are placed between each digit. Example: 4.3.2.1.5.5.5.2.0.2.1. In DNS terms, each digit becomes a zone. Authority can be delegated to any point within the number.
4. A domain (for example, e164.arpa) is appended to the end of the numbers in order to create a FQDN. For example,4.3.2.1.5.5.5.2.0.2.1.e164.arpa.
5. The domain name is queried for the resource records that define URIs necessary to access SIP-based VoIP.

Once the authoritative name server for that domain name is found, ENUM retrieves relevant records and uses that data to complete the call or service. For example, the number 12025551234 returns sip:my.name@bigcompany.com.

About NAPTR Records

ENUM uses NAPTR records for URI resource records. NAPTR records are used to translate E. 164 addresses to SIP addresses. An example of a NAPTR record is:

```
$ORIGIN 4.3.2.1.5.5.5.2.0.2.1.e164.arpa.
IN NAPTR 100 10 "u" "sip+E2U"  "!^.*$!sip:phoneme@example.net!"
```

This example specifies that if you want to use the "sip+E2U" service, you should use sip:phoneme@example.net as the address.

The regular expression can be used by a telephone company to easily assign addresses to all of its clients. For example, if your number is +15554242, your SIP address is sip: 4242@555telco.example.net; if your number is +15551234, your SIP address is sip: 1234@555telco.example.net.

About the Oracle USM ENUM Functionality

The ENUM functionality lets the Oracle USM make an ENUM query for a SIP request. The ENUM lookup capability lets the Oracle USM transform E.164 numbers to URIs during the process of routing (or redirecting) a call. During the routing of a SIP call, the Oracle USM uses a local policy attribute to determine if an ENUM query is required and if so which ENUM server(s) need to be queried. A successful ENUM query results in a URI that is used to continue routing or redirecting the call.

Configurable Lookup Length

You can configure a lookup length in the ENUM configuration that provides for more efficient caching of URI lookup results; in it, you can specify the length of the string for the DNS request starting from the most significant digit. This provides more flexibility for length matching, which is useful given the amount of wild card matching available in ENUM services. Specific ENUM groups might only be intended to provide NPANXX or wild card results.

UDP Datagram Support for DNS NAPTR Responses

The Oracle USM's default behavior is to conform to the DNS standard defined in RFC 1035 Domain Names: Implementation and Specification, which sets a maximum size for UDP responses of 512 bytes. This limitation means that responses larger than 512 bytes are truncated (set with the TC, or truncation, bit). In addition, this limitation protects network and system resources because using TCP consumes an undesirable amount of both.

However, you can configure support ENUM queries that manage larger UDP DNS responses as set out in RFC 2671, Extension Mechanisms for DNS (EDNS0), enabling your Oracle USM to manage responses beyond 512 bytes. According to RFC 2671, senders can advertise their capabilities using a new resource record (OPT pseudo-RR), which contains the UDP payload size the sender can receive. When you specify a maximum response size over 512 bytes, then the Oracle USM add the OPT pseudo-RR to the ENUM query—without which the ENUM server will truncate the response.

Custom ENUM Service Type Support

You can configure the ENUM service type that you want to use for an ENUM group. The Oracle USM has always supported E2U+sip and sip+E2U by default, and still does. With Release S-C6.1.0, however, you are also able to configure the service type to those supported in RFCs 2916 and 3721.

For example, you can now set the service type in the ENUM configuration to support E2U+sip and E2U+voicemsg:sip. When you configure customer ENUM service types on your system, however, you should note the following:

- New entries in the **service-type** parameter overwrite pre-existing values, including the default values.
- Because of the overwriting noted above, you must include the defaults (if you want them configured) when you are adding additional ENUM service type support. That is, you have to also type in E2U+sip and sip+E2U if you want them to be used in addition to the customized types you are setting.

ENUM Failover and Query Distribution

ENUM Query Distribution

The Oracle USM can intelligently distribute ENUM queries among all configured ENUM servers. By setting the enum config's **query method** parameter to round robin, the Oracle USM will cycle ENUM queries, sequentially, among all configured ENUM servers. For example, query 1 will be directed to server 1, query 2 will be directed to server 2, query 3 will be directed to server 3, and so on.

The default query method, hunt, directs all ENUM queries toward the first configured ENUM server. If the first server is unreachable, the Oracle USM directs all ENUM queries toward the next configured ENUM server, and so on.

Failover to New enum-config

When an enum-config's configured servers are unreachable via the network, i.e., no response is received on a query, the Oracle USM can failover to a defined ENUM config that contains different enum servers to query. This failover behavior works when all servers in an enum config are unreachable, rather than when the Oracle USM receives not-found type responses.

The Oracle USM queries each ENUM server once before trying the next configured server, and then ultimately trying the servers listed in the **failover-to** enum config. If the failover-to servers also are unreachable, the Oracle USM fails the call; the failover-to behavior does not recurse among enum-configs, it only checks the first, linked enum-config.

ENUM Server Operation States

After 5 consecutive failed attempts, an ENUM server is considered Out of Service (OOS). All subsequent queries which would be directed to the OOS servers are immediately directed to the first non-OOS server. ENUM servers return to in-service after 600 seconds. If all configured ENUM servers are OOS, the Oracle USM fails the call.

After the first failed attempt to reach an ENUM server, it is placed in a Time Out state, which it stays in for 30 seconds. Within this 30 seconds it will not be contacted when an ENUM query is made. After the 30 seconds pass, the ENUM server goes back to an in-service state.

Server Availability Monitoring

The Oracle USM can probe an ENUM server's health by sending it a standard ENUM NAPTR query and receiving a valid answer. The query is for the phone number defined in the **health query number** parameter, which should be one that the ENUM servers can positively resolve. As long as the query succeeds, that ENUM server maintains its in-service state and is available for ENUM queries. Any lack of response, whether network based (time-outs), or application based (DNS error or not found response) is considered a query failure and the server is set to OOS and unavailable for ENUM queries.

The Oracle USM continuously checks the health of all configured ENUM servers to determine their current state and monitor for failed servers' return to service. All servers are checked for availability at the **health query interval** parameter, as defined in seconds.

> **Note:**
>
> When ENUM server availability monitoring is enabled, ENUM servers can only exist in an in-service or out-of-service states; Without the health query interval defined, server availability monitoring is disabled, and ENUM servers exist in three service states.

ENUM Server IP Address and Port

You can configure an IP address and port for each enum server listed in the enum-servers parameter. IP address and port are specified in XXX.XXX.XXX.XXX:YYYY format with a port value range of 1024-65535. If the port number is not specified, 53 is assumed.

The Oracle USM supports IPv6 ENUM configurations in IPv6 realms. The enumservers parameter in the enum-config configuration parameter can be configured IPv6 addresses in addition to IPv4 addresses. When IPv6 Addresses are used, the realm configured in the realm-id parameter must be an IPv6 realm.

Unapplicable SNMP Traps and Objects

When only IPv4 ENUM servers are configured, all legacy SNMP object and trap functionality remains the same. When IPv6 addressing is used for ENUM servers, these existing SNMP objects are obsoleted.

```
apSysMgmtENUMStatusChangeTrap           NOTIFICATION-TYPE
apENUMServerStatusTable OBJECT-TYPE
```

IPv6 ENUM SNMP Traps and Objects

New SNMP trap notifies operators of ENUM Server Status change.

```
apAppsENUMServerStatusChangeTrap           NOTIFICATION-TYPE
        OBJECTS         { apAppsENUMConfigName,
                          apAppsENUMServerInetAddressType,
                          apAppsENUMServerInetAddress,
```

```
                        apAppsENUMServerStatus }
        STATUS          current
        DESCRIPTION
        " The trap will be generated if the reachability status of an ENUM
        server changes."
        ::= { apAppsNotifications 1 }
```

The following objects support this trap.

```
apAppsENUMServerStatusTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApAppsENUMServerStatusEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "A read-only table to hold the status of configured ENUM servers,
indexed by the name of the enum server, server address type and server IP.
            Please note this table is the replacement of apENUMServerStatusTable
defined in ap-smgmt.mib, where the table was obsoleted."
        ::= { apAppsMIBTabularObjects 1 }
apAppsENUMServerStatusEntry OBJECT-TYPE
        SYNTAX          ApAppsENUMServerStatusEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "An entry designed to hold the status of a single ENUM server"
        INDEX { apAppsENUMConfigName,
                apAppsENUMServerInetAddressType,
                apAppsENUMServerInetAddress }
        ::= { apAppsENUMServerStatusTable 1 }
ApAppsENUMServerStatusEntry ::= SEQUENCE {
        apAppsENUMConfigName                DisplayString,
        apAppsENUMServerInetAddressType     InetAddressType,
        apAppsENUMServerInetAddress         InetAddress,
        apAppsENUMServerStatus              INTEGER
        }
apAppsENUMConfigName    OBJECT-TYPE
        SYNTAX          DisplayString
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The name of the enum-config element that contains this
            ENUM server."
        ::= { apAppsENUMServerStatusEntry 1 }
apAppsENUMServerInetAddressType   OBJECT-TYPE
        SYNTAX          InetAddressType
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The IP address of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 2 }
apAppsENUMServerInetAddress   OBJECT-TYPE
        SYNTAX          InetAddress
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The IP address of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 3 }
apAppsENUMServerStatus  OBJECT-TYPE
        SYNTAX          INTEGER {
                                inservice(0),
                                lowerpriority(1),
```

```
                                oosunreachable(2)
                                }
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
            "The status of this ENUM server."
        ::= { apAppsENUMServerStatusEntry 4 }
apAppsENUMServerStatusGroup  OBJECT-GROUP
        OBJECTS {
                apAppsENUMConfigName,
                apAppsENUMServerInetAddressType,
                apAppsENUMServerInetAddress,
                apAppsENUMServerStatus
                }
        STATUS          current
        DESCRIPTION
                "Report the status of configured ENUM servers."
        ::= { apAppsObjectGroups 1  }
apAppsEnumServerNotificationsGroup NOTIFICATION-GROUP
      NOTIFICATIONS {
                        apAppsENUMServerStatusChangeTrap
                    }
                STATUS          current
                DESCRIPTION
                "A collection of traps to extend reporting capabilities."
      ::= {  apAppsNotificationGroups 1 }
```

Caching ENUM Responses

As DNS responses often lead to further DNS queries, a DNS server can send additional multiple records in a response to attempt to anticipate the need for additional queries. The Oracle USM can locally cache additional NAPRT, SRV, and A records returned from an ENUM query to eliminate the need for unnecessary external DNS requests by enabling the **cache addl records** parameter. These cached records can then be accessed by internal ENUM and DNS agents.

The unprompted NAPTR, SRV, or A record returned to the Oracle USM must include complete information to resolve a call to be added to the local DNS/ENUM cache, otherwise the Oracle USM will preform an external query to fine the address it is looking to resolve.

Cached entries are per ENUM config. That means if one ENUM config has a number of cached entries, and an ENUM request is directed through a different ENUM config, the second configuration is not privy to what the first configuration has cached.

The Oracle USM uses the shorter lifetime of the DNS response's TTL or the server dns attribute's transaction-timeout to determine when to purge a DNS record from the local cache.

Source URI Information in ENUM Requests

ENUM queries can be configured to include the source URI which caused the ENUM request by enabling the **include source info** parameter. The Oracle USM can add the P-Asserted-ID URI (only if not in an INVITE) or the From URI into an OPT-RR Additional Record to be sent to the ENUM server. It can be useful to specify the originating SIP or TEL URI from a SIP request which triggered the ENUM query, so the ENUM server can provide a customized response based on the caller.

This feature implements the functionality described in the Internet Draft, DNS Extension for ENUM Source-URI, draft-kaplan-enum-source-uri-00.

When a P-Asserted-ID is blocked or removed before the ENUM query is made, the Oracle USM only sends the URI in the From header.

Note that to support this feature, according to the Internet draft, ENUM clients must support 1220 bytes in UDP responses. Therefore, if this feature is enabled, and the max response size parameter is not set i.e., with a 512 byte default, the Oracle USM will set the size to 1200 on the OPT-RR records sent.

Operation Modes

There are four modes of ENUM operation that are selected on a global basis:

- stateless proxy
- transaction stateful proxy
- session stateful proxy
- B2BUA with or without media

Stateless Proxy Mode

The stateless proxy mode is the most basic form of SIP operation. The stateless proxy mode:

- Has the least number of messages per call. No record route header is added and there are no 100 Trying or BYEs.
- Does not keep transaction state (timers and retransmission). There are no session counters and no session stop time. No session stop time means no RADIUS STOP records.
- Has no limits on session state.
- Can restrict functionality by specification. This can mean no media management, limited potential for RADIUS accounting, and no CALEA (no Release/BYE messages for CDC).
- Acts primarily as a routing device, with local policy routing and ENUM routing.

Transaction Stateful Proxy

In the transaction stateful proxy mode:

- Adds state to the proxy (not dialogs).
- Has lower number of messages per call. No Record Route header added and no BYES.
- Keeps transaction state (timers and retransmissions.
- Enforces session restrictions (32k) because of state management. These restrictions can be increased.
- Can restrict functionality by specification. This can mean no media management, limited potential for RADIUS accounting, and no CALEA (no Release/BYE message for CDC).
- Acts as routing device with transaction timers, with local policy routing and ENUM routing.
- Can off-load some transactions across unreliable links.

Session Stateful Proxy

The session stateful proxy mode:

- Maintains dialog state as a proxy.
- Includes BYES (though cannot be inserted)
- Keeps transaction state (timers and retransmission)
- Provides per-session information such as session counters per session agent, RADIUS STOP accounting record generation, CALEA CDC generation.
- Enforces session restrictions (32k) because of state management.
- Does not provide media management. There is no CALEA CCC.
- Routes full sessions with transaction timers with local policy routing and ENUM routing.

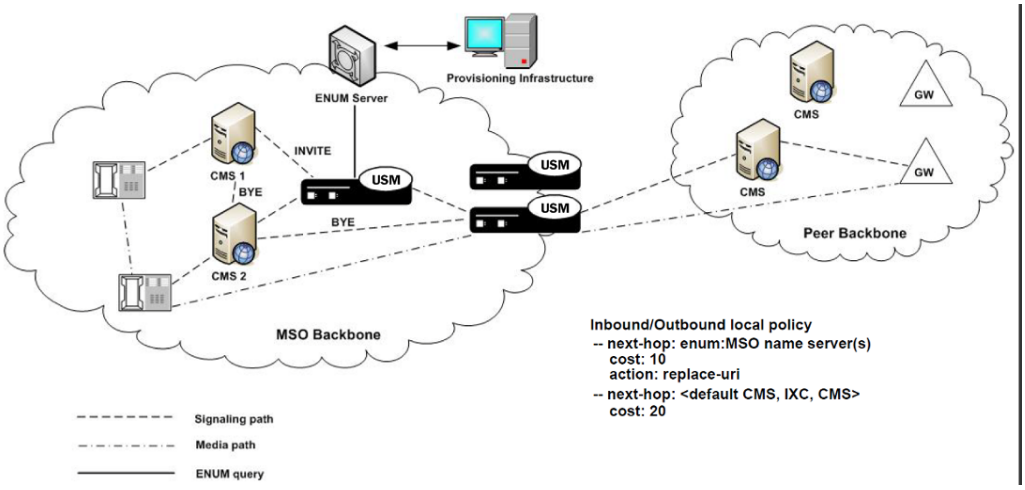
B2BUA

The B2BUA mode:

- Acts as UAS and UAC within call flow.
- Includes BYES (can be inserted).
- Keeps transaction state (timers and retransmissions)
- Provides per-session information such as session counters per session agent, RADIUS STOP accounting record generation, CALEA CDC generation.
- Enforces session restrictions (32k) because of state management.
- Can provide media management, including media routing through a single IP address with topology masking, CALEA CCC, media watchdogs for state management.
- Routes full sessions with topology masking. Includes rewriting Via, Route, Contact headers, full NATing with SIP NAT or header manipulation, direct bridging, local policy routing, and ENUM routing.

Example ENUM Stateless Proxy

The following diagram shows the Oracle USM using ENUM to query a local subscriber database. The Oracle USM serves as the inbound and outbound routing hub and performs media management. Calls are routed throughout the MSO network using ENUM lookup results.



ENUM Configuration

This section shows you how to configure ENUM on your Oracle USM.

To configure ENUM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **enum-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)#
   ```

4. **name**—Enter a string that uniquely identifies this ENUM configuration. You use this name in other areas of the Oracle USM configuration to refer to this ENUM configuration. For example, in the local policy attributes.

5. **top-level-domain**—Enter the domain extension to be used when querying the ENUM servers for this configuration. For example, e164.arpa. The query name is a concatenation of the number and the domain.

   For example the number is +17813334444 and the domain is e164.arpa, the query name would be 4.4.4.4.3.3.3.1.8.7.1.e164.arpa.com.

6. **realm-id**—Enter the realm where the ENUM servers can be reached. The realm ID is used to determine on which network interface to issue the ENUM query.

7. **enum-servers**—Enter the list of ENUM servers (an ENUM server and corresponding redundant servers) to be queried. Separate each server address with a space and enclose list within parentheses.

   The first server on this list is the first one to be queried. If the query times out (including retransmissions) without getting a response, the next server on the list is queried and so on.

8. **service-type**—Enter the ENUM service types you want supported in this ENUM configuration. Possible entries are E2U+sip and sip+E2U (the default), and the types outlines in RFCs 2916 and 3721.

   This parameter defaults to the following service types: E2U+sip and sip+E2U.

   You can enter multiple services types in the same entry, as in this example:

   ```
   ORACLE(enum-config)# service-type E2U+sip,sip+E2U,E2U+voicemsg
   ```

9. **query-method**—Set the strategy the Oracle USM uses to contact ENUM servers. Valid values are:

   - hunt—Directs all ENUM queries toward the first configured ENUM server. If the first server is unreachable, the Oracle USM directs all ENUM queries toward the next configured ENUM server, and so on.
   - round-robin—Cycles all ENUM queries, sequentially, among all configured in-service ENUM servers. Query 1 will be directed to server 1, query 2 will be directed to server 2, query 3 will be directed to server 3.

10. **timeout**—Enter the total time in seconds that should elapse before a query sent to a server (and its retransmissions) will timeout. If the first query times out, the next server is queried and the same timeout is applied. This process continues until all the servers in the list have timed out or until one of the servers responds.

    The retransmission of ENUM queries is controlled by three timers. These timers are derived from this timeout value and from underlying logic that the minimum allowed retransmission interval should be 250 milliseconds; and that the Oracle USM should retransmit 3 times before timing out to give the server a chance to respond. The valid values are:

    - **Init-timer**—Is the initial retransmission interval. If a response to a query is not received within this interval, the query is retransmitted. To safeguard from performance degradation, the minimum value allowed for this timer is 250 milliseconds.
    - **Max-timer**—Is the maximum retransmission interval. The interval is doubled after every retransmission. If the resulting retransmission interval is greater than the value of max-timer, it is set to the max-timer value.
    - **Expire-timer**—Is the query expiration timer. If a response is not received for a query and its retransmissions within this interval, the server will be considered non-responsive and the next server in the list will be tried.

      The following examples show different timeout values and the corresponding timers derived from them.

      timeout >= 3 seconds

      ```
      Init-timer = Timeout/11
      Max-Timer = 4 * Init-timer
      Expire-Timer = Timeout
      ```

      timeout = 1 second

      ```
      Init-Timer = 250 ms
      Max-Timer = 250 ms
      Expire-Timer = 1 sec
      ```

      timeout = 2 seconds

      ```
      Init-Timer = 250 ms
      Max-Timer = 650 ms
      Expire-Timer = 2sec
      ```

11. **cache-inactivity-timer**—Enter the time interval in seconds after which you want cache entries created by ENUM requests deleted, if inactive for this interval. If the cache entry gets a hit, the timer restarts and the algorithm is continued until the cache entry reaches its actual time to live.

    Setting this value to zero disables caching. For optimal performance, set this to one hour. Rarely used cache entries are purged and frequently used entries are retained. The default value is **3600**. The valid range is:

    - Minimum—0
    - Maximum—999999999

12. **lookup-length**—Specify the length of the ENUM query, starting from the most significant digit. The default is **0**. The valid range is:

    - Minimum—1
    - Maximum—255

13. **max-response-size**—Enter the maximum size in bytes for UDP datagrams in DNS NAPTR responses. This parameter takes values from 512 (default) to 65535. Although the maximum value you can set is 65535, Oracle recommends configuring values that do not exceed 4096 bytes.
14. **health-query-number**—Set this parameter to a standard ENUM NAPTR query that will consistently return a positive response from the ENUM server.
15. **health-query-interval**—Set this parameter to the number of seconds to perpetually probe ENUM servers for health.
16. **failover-to**—Set this parameter to the name of another ENUM-config which to failover to under appropriate conditions.
17. **cache-addl-records**—Set this parameter to **enabled** for the Oracle USM to add additional records received in an ENUM query to the local DNS cache.
18. **include-source-info**—Set this parameter to enabled for the Oracle USM to send source URI information to the ENUM server with any ENUM queries.
19. Save your work.

Example

The following example shows an ENUM configuration called enumconfig.

```
enum-config
        name                            enumconfig
        top-level-domain
        realm-id                        public
        enum-servers                    10.10.10.10:3456
                                        10.10.10.11
        service-type                    E2U+sip,sip+E2U
        query-method                    hunt
        timeout                         11
        cacheInactivityTimer            3600
        max-response-size               512
        health-query-number             +17813245678
        health-query-interval           0
        failover-to                     enumconfig2
        cache-addl-records              enabled
        include-source-info             disabled
```

Configuring the Local Policy Attribute

You can specify that an ENUM query needs to be done for the routing of SIP calls. You do so by configuring the local policy's next-hop attribute with the name of a specific ENUM configuration, prefixed with the enum: tag. For example: enum:test

You can configure multiple next-hops with different ENUM servers or server groups (possibly with different top-level-domains). If the first ENUM server group you enter as the next hop is not available, one of the others can be used.

> **Note:**
>
> A new parameter called action has replaced the policy attribute's replace-uri parameter available prior to build 211p19.
> To configure local policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. **next-hop**—Enter the name of the ENUM configuration with the prefix enum:. For example, enum:test.
5. **action**—Set to **redirect** if you want to send a REDIRECT message back to the calling party with the information returned by ENUM in the Contact. The calling party then needs to send a REDIRECT using that information. The default value is **none**. Valid values are:
   - **none**—No specific actions requested.
   - **replace-uri**—To replace the next Request-URI with the next hop.
   - **redirect**—To send a redirect response with this next hop as contact.
6. Save and activate your configuration.

Local Policy Example

The following example shows one local policy with the next-hop configured to use enum:test and a second with the next-hope configured to use enum:test_alternate.

```
local-policy
     from-address            *
     to-address              *
     source-realm            public
     activate-time           N/A
     deactivate-time         N/A
     state                   enabled
     last-modified-date      2006-03-09 09:18:43
     policy-attribute
       next-hop                enum:test
       realm                   public
       action                  none
       terminate-recursion     disabled
       carrier
       start-time              0000
       end-time                2400
       days-of-week            U-S
       cost                    1
       app-protocol            SIP
       state                   enabled
```

```
    media-profiles
  policy-attribute
    next-hop               enum:test_alternate
    realm                  public
    action                 none
    terminate-recursion    disabled
    carrier
    start-time             0000
    end-time               2400
    days-of-week           U-S
    cost                   2
    app-protocol           SIP
    state                  enabled
```

CNAM Subtype Support for ENUM Queries

CNAM, calling name, data is a string up to 15 ASCII characters of information associated with a specific calling party name. The *Internet-draft, draft-ietf-enum-cnam-08.txt*, registers the Enumservice 'pstndata' and subtype 'cnam' using the URI scheme 'pstndata:' to specify the return of CNAM data in ENUM responses. The Oracle USM recognizes CNAM data returned via this mechanism. CNAM data is then inserted into the display name of the From: header in the original Request. If a P-Asserted-ID header is present in the original request, the CNAM data is inserted there as well.

CNAM data is identified by an ENUM response with service-type: E2U+pstndata:cnam

CNAM support is configured in the sip profile configuration element, which can then be applied to either a session agent, realm, or SIP interface.

The Oracle USM can preform CNAM queries on the signaling message's ingress or egress from the system by setting the cnam lookup direction parameter to either ingress or egress. If the CNAM lookup direction parameters are configured on both the ingress and egress sides of a call, the Oracle USM will only preform the lookup on the ingress side of the call.

CNAM Unavailable Response

A CNAM response can include a Calling Name Privacy Indicator parameter ('unavailable=p') or Calling Name Status Indicator parameter ('unavailable=u') in responses. The Oracle USM can insert a custom reason string into the SIP message's From and P-Asserted-ID header in the original requires.

Configuring the **cnam unavailable ptype** parameter inserts the specified text into the From and P-Asserted-ID headers when a CNAM response contains the unavailable=p parameter.

Configuring the **cnam unavailable utype** parameter inserts the specified text into the From and P-Asserted-ID headers when a CNAM response contains the unavailable=u parameter.

SIP Profile Inheritance

CNAM features, via the SIP Profile configuration element can be applied to session agents, realms, and SIP interfaces. The more generalized object inherits the more specific object's values. For example, if CNAM support via a SIP profile is configured on a session agent, the expected processing will override any SIP profile configuration on the downstream realm or SIP interface. Likewise, if CNAM support is unconfigured on the receiving session agent, but configured in the realm, CNAM configuration on the SIP interface will be ignored.

CNAM Subtype Support Configuration

To enable the Oracle USM to preform CNAM subtype ENUM queries, you must configure a SIP profile with an enum-config object (that points to valid ENUM servers). The referenced enum-config configuration element lists the servers to contact for CNAM type queries (and other general ENUM server interaction parameters).

To configure CNAM subtype support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **name**—Enter a string that uniquely identifies this SIP profile configuration. You use this name in other areas of the Oracle USM configuration to refer to this SIP profile in session agents, realms, or SIP interfaces.
5. **cnam-lookup-server**—Set this parameter to the name of an ENUM-config to that will query ENUM servers for CNAM data.
6. **cnam-lookup-dir**—Set this parameter to **ingress** or **egress** to identify where the Oracle USM performs a CNAM lookup with respect to where the call traverses the system. The default value is **egress**.
7. **cnam-unavailable-ptype**—Set this parameter to a string, no more than 15 characters, to indicate that the unavailable=p parameter was returned in a CNAM response.
8. **cnam-unavailable-utype**—Set this parameter to a string, no more than 15 characters, to indicate that the unavailable=u parameter was returned in a CNAM response.
9. Save your work.

Using the Local Route Table (LRT) for Routing

The LRT allows the Oracle USM to determine next hops and map E.164 to SIP URIs locally for routing flexibility.

The LRT uses a local route cache that is populated by a local XML file on the Oracle USM. Each local cache is populated from one defined XML file. For routing, the local route cache operates in a way similar to the ENUM model where a local policy next hop specifies the local route table that the Oracle USM references. For example, you can configure one next hop to use one table, and another next hop to use a different table.

Similar to the ENUM model, the Oracle USM typically performs a local route table lookup using the telephone number (TN) of the SIP Request-URI. This is the user portion of the URI, and the Oracle USM ignores user parameters or non-digit characters. The local route table XML file defines the matching number and the resulting regular expression replacement value such as ENUM NAPTR entries do. The Oracle USM uses the resulting regular expression to replace the Request-URI, and it uses the hostname or IP address portion to determine the next

hop. If the hostname or IP address matches a configured session agent, the request is sent to that session agent. If the Oracle USM does not find a matching session agent for the hostname/IP address, the Oracle USM either performs a DNS query on the hostname to determine its IP address or sends the request directly to the IP address.

When the next hop is defined as a user-parameter lookup key, such as a routing number (RN) or carrier identification code (CIC), the defined key is used for the local route table lookup.

The Oracle USM can attempt up to 10 next hops per LRT entry in the order in which they appear in the XML file. If the next hop is unsuccessful, the Oracle USM tires the next hop on list. An unsuccessful hop may occur when an out-of-service session agent or the next hop responds with a failure response.

> **Note:**
>
> Entering XML comments on the same line as LRT XML data is not supported.

The Oracle USM can perform local route table lookups for SIP requests and communicate the results to the SIP task. The new task processes the new local routing configuration objects.

When a SIP call is routed, the Oracle USM uses local policy attributes to determine if a local route table lookup is required. If a lookup is needed, the Oracle USM selects the local routing configuration to use. Successful local route table lookups result in URIs that can be used to continue routing and redirecting calls.

Local Route Table (LRT) Performance

Capabilities

- Loads approximately 500 LRT tables during boot time
- Loads 100,000 entries per LRT file
- Loads 2,000,000 LRT entries total per system

Constraints

- You cannot configure the Oracle USM with 500 LRT files each with 100,000 entries.
- Actual performance that affects the interaction among the three performance attributes varies with system memory and configuration.

Local Routing Configuration

This section shows you how to:

- Set up local route configuration
- Specify that a set of local policy attributes needs to use local routing

Configure Local Routing

The local routing configuration is an element in the ACLI session-router path, where you configure a name for the local route table, the filename of the database corresponding to this table, and the prefix length (significant digits/bits) to be used for lookup.

To configure local routing:

1. In Superuser mode, type **configure terminal**, and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router**, and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config,**and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **name**—Enter the name (a unique identifier) for the local route table; this name is used for reference in the local policy attributes when to specify that local routing should be used. There is no default for this parameter, and it is required.

5. **file-name**—Enter the name for the file from which the database corresponding to this local route table will be created. You should use the .gz format, and the file should be placed in the /code/lrt/ directory. There is no default for this parameter and it is required.

6. **prefix-length**—Enter the number of significant digits/bits to used for lookup and cache storage. The default value is **0**. The valid range is:

   - Minimum—0
   - Maximum—999999999

7. Save and activate your configuration.

   The following example displays a typical local routing configuration.

   ```
   local-routing-config
           name                            lookup
           file-name                       abc.xml.gz
           prefix-length                   3
   ```

Applying the Local Routing Configuration

Apply the local routing configuration by calling it to use in the local policy attributes. You do this by setting a flag in the **next-hop** parameter along with the name of the local routing configuration that you want to use.

To apply the local routing configuration:

1. In Superuser mode, type **configure terminal**, and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router**, and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-policy,** and press Enter.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes,** and press Enter.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

5. **next-hop**—In the **next-hop** parameter, type in **lrt:** followed directly by the name of the local routing configuration to be used. The **lrt:** tag tells the Oracle USM that a local route table will be used.

```
ACMEPACKET(local-policy-attributes)# next-hop lrt:lookup
```

6. Save and activate the configuration.

LRT Entry Matching

When searching an LRT for a matching route, the Oracle USM can be configured with one of three match modes with the match mode parameter in the local routing config. These modes are:

- exact—When searching the applicable LRT, the search and table keys must be an exact match.
- best—The longest matching table key in the LRT is the chosen match.
- all—The all mode makes partial matches where the table's key value is a prefix of the lookup key. For example, a lookup in the following table with a key of 123456 returns entries 1, 3, and 4. The 'all' mode incurs a performance penalty because it performs multiple searches of the table with continually shortened lookup keys to find all matching entries. This mode also returns any exact matches too.

Entry#	Key	Result
1	1	<sip:\0@host1.example.com>
2	122	<sip:\0@host22.example.com>
3	123	<sip:\0@host3.example.com>
4	1234	<sip:\0@host4.example.com>
5	1234567	<sip:\0@host7.example.com>
6	1235	<sip:\0@host5.example.com>

LRT Entry Matching Configuration

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **match-mode**—Set this parameter to either **best**, **all**, or leave it as **exact** which is the default. This indicates to the Oracle USM how to determine LRT lookup matches.
5. Save your work using the **done** command.

LRT String Lookup

The Oracle USM can search an LRT for either E.164 or string table keys. This selection is on a global basis. When the string-lookup parameter is **disabled** (default) in the local routing configuration, all lookups with be E.164 type, except when:

- If eloc-str-lkup is enabled in a matching local policy's policy-attribute, E-CSCF procedures are applied and the resulting lookup type is 'string'.

- The Oracle USM also performs string lookups exclusively when a compound lookup key is specified.

When the lookup type is 'E.164', the lookup is skipped if the lookup key is not a valid telephone number (i.e. it must contain only digits).

LRT String Lookup Configuration

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **string-lookup**—Set this parameter to **enabled** for the Oracle USM to perform LRT lookups on table keys of a string data type. Leave this parameter to its default as disabled to continue using E.164 type lookups.

5. Save your work using the **done** command.

Directed Egress Realm from LRT ENUM

A message can be sent into a specific egress realm identified in an ENUM query or LRT lookup. The egress realm is noted by a configurable parameter in the result URI. The Oracle USM is configured with the name of this parameter, that indicates an egress realm name, and looks for it in the returned URI.

To configure the parameter name, the egress-realm-param option is added to the sip config using the following format:

```
egress-realm-param=<name>
```

Where <name> is the parameter name to extract the egress realm name from.

When the egress realm param is defined, the ENUM and LRT results will always be checked for the presence of the URI parameter. The sip config options will apply for received SIP requests.

For example, if egress-realm-param=egress is added to the sip config, a matching entry in the LRT that specifies the egress realm core will look like this:

```
<route>
<user type="E164">+17815551212</user>
<next type="regex">!^.*$!sip:\0@core.example.com;egress=core!</next>
</route>
```

If the URI does not contain the parameter or the parameter identifies a realm that is not configured on the system, the egress realm that is normally applicable (from local policy, SIP-NAT, or session-agent data) will be used.

Directed Egress Realm Configuration

To add an egress parameter to look for in a sip-config:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **egress-realm-param**—Configure this option with the parameter to parse for in a returned ENUM or LRT result: For example

   - **options egress-realm-param=egress**

     In order to append the new option to the sip-config's options list, you must prepend the new option with a plus sign. For example:

     ```
     ORACLE(sip-config)# options +egress-realm-param=egress
     ```

5. Save your work using the ACLI **done** command.

> **Note:**
>
> The egress-realm-param option can be configured similarly in the h323-config.

SIP Embedded Route Header

The Oracle USM examines the ENUM and LRT lookup result for embedded Route headers. In the LRT or as returned in an ENUM query a URI including an embedded route header would look like:

```
<sip:user@example.com?Route=%3Csip:host.example.com;lr%3E>
```

Using embedded Route headers is the Oracle USM's default behavior. This can be overridden by adding the sip-config option use-embedded-route.

When the ENUM or LRT result becomes the top Route header, any embedded Route headers extracted are inserted just after that top Route (which will always be a loose route and include the "lr" URI parameter). In this case, the request will be sent to the top Route.

When the ENUM or LRT results become the Request-URI, any embedded Route headers extracted from the result are inserted before any Route headers already in the outgoing request. After that, if there are any Route headers in the outgoing request and the top Route header has an "lr" URI parameter, the request is sent to the top Route header. Otherwise, the request is sent to the Request URI.

SIP Embedded Route Header Configuration

To set the Oracle USM's default behavior of using embedded route headers from ENUM queries or LRT lookups:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. If you are adding this feature to a pre-existing SIP configuration, you will need to select and edit it.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **use-embedded-route**—Configure this as an option with one of the following arguments:
   - **all** = use embedded routes from both ENUM and LRT results (default)
   - **none** = do not use embedded routes
   - **enum** = use embedded routes from ENUM results only
   - **lrt** = use embedded routes from LRT results only

     In order to append the new option to the sip-config's options list, you must prepend the new option with a plus sign. For example:

     Set the options parameter by typing **options**, a Space, the option name **use-embedded-route**, and then press Enter.

```
ORACLE(sip-config)# options +use-embedded-route=none
```

5. Save your work using the ACLI **done** command.

LRT Lookup Key Creation

This section describes the Oracle USM's LRT lookup key creation capability.

Arbitrary LRT Lookup Key

In addtion to the standard From, To, and and P-Asserted-Identity header fields the Oracle USM can now use the values from any arbitrary SIP header as an LRT or ENUM lookup key. This is preformed by prepending a dollar sign $ by the header name whose value's userinfo portion of the URI will be used as the lookup key. For example, key=$Refer-To would use the userinfo portion of the URI in the Refer-To header of the request as the lookup key.

An ampersand & followed by a header name will use the whole value of the header as the lookup key. For example, key=&X-Route-Key would use the whole value of the X-Route-Key as the lookup key. As a shortcut, an ampersand is not required for a "hidden" header. For example, "key=@LRT-Key" would use the value of the @LRT-Key header as the lookup key.

Hidden Headers for HMR and LRT lookup

When an LRT lookup key is more complex than just the URI's userinfo or a Tel-URI, HMR can be used to extract the data and build a special header.

By using a header name that begins with the at-sign "@" (e.g. @lrt-key), the header can be hidden and not included in outgoing SIP message, thus eliminating the need for an extra HMR rule to remove it.

Since '@' is not a valid character in a header name as defined by RFC 3261, there is no possibility of a collision between a header name defined in the future and a hidden header name beginning with @.

Compound Key LRT Lookup

LRT lookup keys can be combinations of more than one key value. For example, "key= $FROM,$TO" would construct a compound key with the userinfo of the From URI followed by a comma flowed by the userinfo of the To URI.

If the request message contained:

```
From: <sip:1234@example.com>
To: <sip:5678@example.com>
```

The compound key to match this From/To pair is "1234,5678".

In the table lookup, the compound key is a single key value and there is no special treatment of the comma in key matching. The comma is simply an ordinary additional character that is matched like any letter or digit (i.e. the comma must appear in the LRT entry's "type" element data). For example, if the table were configured for "best" match-mode, the lookup key "1234,5678" would match a table entry of "1234,567", but it would not match a table entry of "123,5678".

Retargeting LRT ENUM-based Requests

Request re-targeting is when a target or a request as indicated in the Request-URI, is replaced with a new URI.

The happens most commonly when the "home" proxy of the target user replaces the Request-URI with the registered contact of that user. For example, the original request is targeted at the Address-of-Record of bob (e.g. sip: bob@example.net). The "home" proxy for the domain of the original target, example.net, accesses the location service/registration database to determine the registered contact(s) for the user (e.g. sip:bob@192.168.0.10). This contact was retrieved in a REGISTER request from the user's UA. The incoming request is then re-targeted to the registered contact. When retarget-requests is **enabled**, or the original Request-URI is the Oracle USM itself, the URI from the LRT lookup is used as the new Request-URI for the outgoing request.

When a request is routed rather than re-targeted, the Request-URI is not changed, but one or more Route headers may be inserted into the outgoing request. Sometimes a request which already contains Route headers will be routed without adding additional Route headers.

When the Oracle USM routes requests and the original Request-URI was not the Oracle USM itself, the URI from the LRT /ENUM lookup is added as the top Route: header including the "lr" parameter. The Request-URI then remains unchanged.

Whether the Oracle USM re-targets or routes a request depends on the following:

- The target (Request-URI) of the received request
- The presence of Route headers
- Local Policy Attributes,
- Registration Cache matching.

If the original target is the Oracle USM itself (i.e. the Request-URI contains the IP Address in the SIP interface the request was received on), the request is always retargeted. When the

original target is not the Oracle USM and Local Policy is applied, the request will be re-targeted when the policy attribute action parameter is **replace-uri**. The request will also be re-targeted when the policy attribute specifies an ENUM or LRT lookup.

Retargetting requests can be configured in either the ENUM or LRT config depending on the request URI retrieval method choosen.

Re-targeting LRT ENUM-based Requests Configuration

This section shows you how to configure the Oracle USM to retarget/re-route request message when performing an LRT or an ENUM lookup.

To configure the Oracle USM to retarget or route request messages when performing an LRT lookup:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# local-routing-config
   ORACLE(local-routing-config)#
   ```

4. **retarget-requests**—Leave this parameter set to **enabled** for the Oracle USM to replace the Request-URI in the outgoing request. Change this parameter to **disabled** for the Oracle USM to route the request by looking to the Route header to determine where to send the message.

5. Save your work using the **done** command.

   To configure the Oracle USM to retarget or route request messages when performing an ENUM lookup:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

8. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)#
   ```

9. **retarget-requests**—Leave this parameter set to **enabled** for the Oracle USM to replace the Request-URI in the outgoing request. Change this parameter to **disabled** for the Oracle USM to route the request by looking to the Route header to determine where to send the message.

10. Save your work using the **done** command.

Recursive ENUM Queries

If the Oracle USM receives an A-record in response to an ENUM query, it will reperform that ENUM query to the server received in the A-record.

If the Oracle USM receives an NS record in response to an ENUM query, it will resend the original ENUM query to the DNS server defined in the realm of the FQDN in the NS record. It will use the response to perform a subsequent ENUM query.

This behavior is configured by setting the **recursive query** parameter in the enum config to **enabled**.

Recursive ENUM Queries Configuration

To configure the Oracle USM to query a DNS server for a hostname returned in an ENUM lookup result:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-routing-config** and press Enter.

   ```
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)#
   ```

4. **recursive-query**—Set this parameter to **enabled** for the Oracle USM to query a DNS server for a hostname returned in an ENUM result.
5. Save your work using the **done** command.

Multistage Local Policy Routing

Multistage local policy routing enables the Oracle USM to perform multiple stages of route lookups where the result from one stage is used as the lookup key for the next routing stage.

Routing Stages

A routing stage signifies a re-evaluation of local policy based on the results of a local policy lookup. In the simplest, single stage case, the Oracle USM performs a local policy lookup on a SIP message's Request URI. The result of that local policy lookup is a next hop FQDN, IP address, ENUM lookup, or LRT lookup; that result is where the Oracle USM forwards the message. In the multistage routing model, that resultant next hop is used as the lookup key for a second local policy lookup.

The results of each stage do not erase the results of the previous stage. Thus, previous results are also possible routes to use for recursion, but the next stage results are tried first.

> **Note:**
>
> Setting a next hop to a SAG in a multistage scenario constitutes an error.

Multi-stage Routing Source Realm

By default, the Oracle USM uses the realm within which a message was received as the source realm through all stages of a multistage local policy routing lookup. You can change this by setting the **multi-stage-src-realm-override** parameter in the session router config to enabled.

Enabling this setting causes the Oracle USM to use the next-hop realm from the current local policy stage as the source realm for the next stage of the lookup. This source realm selection process also repeats for each stage of a multistage routing scenario.

Network Applications

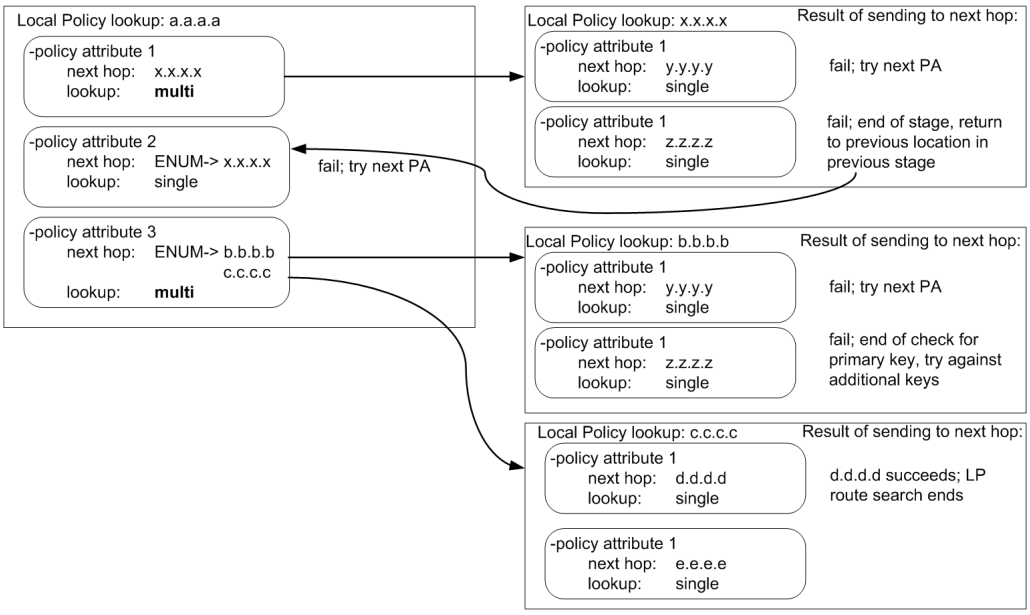
The following are typical applications of multistage routing:

- An operator might need to query an ENUM server for a destination number. Based on the NAPTR result of the ENUM query, the Oracle USM performs a local policy lookup to decide how to route the request, perhaps based on a LRT table lookup.
- An operator might need to query one ENUM server for a number portability lookup, then based on the routing number perform a second ENUM query to a different server to learn which carrier to use for the routing number. Then, then based on the identified carrier perform a LRT lookup for what next-hop(s) to use for that carrier.
- An operator might query an LRT table to confirm the allowed source number. Then, based on the result, query an ENUM server for destination routing.

Multistage Routing Conceptual Example

Multistage routing is enabled by setting a policy attribute's lookup parameter to multi. Instead of replacing the SIP message's request URI with the policy attribute's next hop address or response from an ENUM or LRT lookup, the system uses that next hop or ENUM or LRT lookup response to reconstruct the SIP message. The reconstructed SIP message is fed again through all configured local policy configuration elements (and policy attribute sub elements). Each time the Oracle USM re-evaluates a SIP message against local policies, it is considered an additional routing stage. When multiple records are returned from an ENUM or LRT lookup, the Oracle USM evaluates the first response against all applicable local policies. If unsuccessful, the Oracle USM evaluates all additional responses, in turn, against all applicable local policies.

For example:



Multistage Routing Example 2

The following three local policy configuration elements are configured in the Oracle USM:

```
Local Policy 1
    from-addres     *
    to-address      159
    source-realm    private

    policy-attribute
        next-hop    lrt:default-lrt
        lookup      multi
    policy-attribute
        next-hop    192.168.200.50
        lookup      single
```

```
Local Policy 2
    from-addres     *
    to-address      192.168.1.49
    source-realm    private

    policy-attribute
        next-hop    lrt:carrier-lrt
        lookup      multi
    policy-attribute
        next-hop    lrt:emergency
        lookup      single
```

```
Local Policy 3
    from-addres     *
    to-address      215680000002
    source-realm    private

    policy-attribute
        next-hop    192.168.200.98
        lookup      single
    policy-attribute
        next-hop    192.168.200.97
        lookup      single
    policy-attribute
        next-hop    192.168.200.44
        lookup      multi
```

```
<route>
    <user type="E164">159</user>
        <next type="regex">!^.*$!sip:11568000000@192.168.200.47!</next>
        <next type="regex">!^.*$!sip:215680000002@192.168.200.99!</next>
        <next type="regex">!^.*$!sip:11578000000@192.168.200.44!</next>
</route>
```

```
INVITE sip:159@192.168.1.49:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.48:5060
From: sipp <sip:sipp@192.168.1.48:5060>;tag=1
To: sut <sip:159@192.168.1.49:5060>
Call-ID: 1-4576@192.168.1.48
CSeq: 1 INVITE
Contact: sip:sipp@192.168.1.48:5060
Max-Forwards: 70
Subject: Performance Test
Content-Type: application/sdp
Content-Length:  135
```

The local route table in default-lrt appears as follows:

```
<route>
    <user type="E164">159</user>
                <next type="regex">!^.*$!sip:11568000000@192.168.200.47!</next>
        <next type="regex">!^.*$!sip:215680000002@192.168.200.99!</next>
        <next type="regex">!^.*$!sip:11578000000@192.168.200.44!</next>
</route>
```

1. The Oracle USM receives an INVITE on realm, private (SDP is omitted below):

   ```
   INVITE sip:159@192.168.1.49:5060 SIP/2.0
   Via: SIP/2.0/UDP 192.168.1.48:5060
   From: sipp <sip:sipp@192.168.1.48:5060>;tag=1
   To: sut <sip:159@192.168.1.49:5060>
   Call-ID: 1-4576@192.168.1.48
   CSeq: 1 INVITE
   Contact: sip:sipp@192.168.1.48:5060
   Max-Forwards: 70
   Subject: Performance Test
   Content-Type: application/sdp
   Content-Length:  135
   ```

2. The Oracle USM performs a local policy search based on the following parameters:

   ```
   from-address:    sipp <sip:sipp@192.168.1.48:5060>;tag=1
   to-address:            sip:159@192.168.1.49:5060
   Source Realm:    private
   ```

3. The local policy search returns the four following routes to try:

   ```
   lrt:default-lrt
   192.168.200.50
   lrt:emergency
   lrt:carrier-lrt
   ```

The first next-hop route will be an LRT query. In addition, this policy attribute is configured with lookup=multi, meaning the results of the LRT query should be used for another local policy query, i.e., a second stage. More specifically, the request-uri that was received in response to the LRT query will be used as the to-uri in the next LP query.

The Oracle USM performs the LRT lookup in the default-lrt configuration element and is returned the following:

```
sip:11568000000@192.168.200.47
sip:215680000002@192.168.200.99
sip:11578000000@192.168.200.44
```

The Oracle USM attempts to use the results from the LRT query for the next stage Local Policy lookup(s). Beginning with the first route and continuing in sequential order, the Oracle USM will try to route the outgoing INVITE message by performing additional Local Policy lookups on the remaining LRT query results, until the INVITE is successfully forwarded.

The Oracle USM performs a local policy query on:

```
sip:11568000000@192.168.200.47
```

Which equates to a local policy lookup on:

```
        from-URI=sipp <sip:sipp@192.168.1.48:5060>;
        to-URI=sip:11568000000@192.168.200.47
Source Realm: private
```

The query fails because there is no Local Policy entry for 11568000000.

The Oracle USM performs a second query on request-uri

```
sip:215680000002@192.168.200.99
```

Which equates to a local policy lookup on:

```
        from-URI=sipp <sip:sipp@192.168.1.48:5060>;
        to-URI=sip:215680000002@192.168.200.99
Source Realm: private
```

The LP query is successful and returns the following next- hops:

```
192.168.200.98
    192.168.200.99
192.168.200.44
```

The three routes shown above represent the next stage of the multistage routing for this INVITE. The policy attributes' lookup parameter is set to single for these next-hops. Therefore, the Oracle USM will attempt to send the outgoing INVITE message to one or more of these next-hops; there are no more stages to check.

4. The Oracle USM sends an INVITE to 192.168.200.98:

```
INVITE sip:215680000002@192.168.200.98;lr SIP/2.0
Via: SIP/2.0/UDP 192.168.200.49:5060
From: sipp <sip:sipp@192.168.1.48:5060>
To: sut <sip:159@192.168.1.49:5060>
Call-ID: SDnhae701-76e8c8b6e168958e385365657faab5cb-v3000i1
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.200.49:5060;transport=udp>
Max-Forwards: 69
Subject: Performance Test
```

```
Content-Type: application/sdp
Content-Length: 140
```

5. If the INVITE is sent to 192.168.200.98 successfully, the local policy routing will conclude and the call will continue processing. Otherwise the Oracle USM will try the other next hops until a route succeeds or all next-hops have been exhausted

Customizing Lookup Keys

When the **next hop** parameter points to perform an ENUM or LRT lookup, it can be provisioned with a "key=" attribute in order to specify a parameter other than the username to perform the lookup on. The following table lists the header, key value, and corresponding syntax to configure the Oracle USM with.

Username from Header:	Key Value	Example
To-URI	$TO	key=$TO
From-URI	$FROM	key=$FROM
P-Asserted-Identity	$PAI	key=$PAI

For a subsequent stage in multistage local policy routing, the lookup key to use for the next stage can be explicitly specified by configuring the **next key** parameter. By default, multistage lookups use the modified Request-URI returned from the ENUM/LRT response as the to-address key for the next local policy lookup. When the **next key** parameter is configured, its value will be used for the to-address key in the subsequent local policy lookup regardless if an ENUM or LRT lookup is configured for that policy attribute. The key syntax is for this parameter is the same as with the Routing-based RN and CIC feature.

Multistage Routing Lookup Termination

It is important for the Oracle USM to have a mechanism to stop performing additional stages of route lookups and limit the number of attempts and results to be tried. Routing termination can be performed at in the non-multistage way or at the global session router level.

Global Local Policy Termination

The Oracle USM can be configured to limit local policy lookups based several aspects of the route lookup process:

- Limiting the number of stages per message lookup—The Oracle USM can limit to the number of additional local policy lookup stages it will perform received message to a maximum of 5. This is configured with the **additional lp lookups** parameter. Leaving this parameter at its default value of 0 essentially disables multistaged local policy lookups.
- Limiting the number of routes per Local Policy lookup—The Oracle USM can limit the number of route results to use as returned for each Local-Policy lookup. This is configured with the **max lp lookups routes per lookup** parameter. Leaving this parameter at its default value of 0 places no limit on the number of returned routes the Oracle USM can try.
- Limiting the total number of routes for all local policy lookups per message request—The Oracle USM can limit the number of route returned in total across all lookups for a given request, including additional stages. This is configured with the **total lp routes** parameter. Leaving this parameter at its default value of 0 places no limit on the number of returned routes the Oracle USMcan try. This parameter overrides any configured options.

Additionally, the Oracle USM monitors for local policy lookup loops which could cause a significant deterioration in performance. If a loop is found, the Oracle USM stops trying the looping route list and proceeds to try any remaining routes..

Multistage Local Policy Routing Configuration

To set up your local policy attributes for routing using the TO header:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit a local policy.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

5. **next-hop**—This is the next signaling host and/or object to query. This parameter can be configured as an IP address, ENUM server, or LRT. You can also add a lookup key to an ENUM server or LRT lookup with the following syntax:

   ```
   next-hop    enum:ENUM-object;key=$TO
   ```

6. **terminate-recursion**—Set this parameter to **enabled** to terminate local policy route recursion when the current stage completes.

7. **lookup**—Leave this parameter at the default **single** for single stage local policy routing or set it to **multi** to enable multistage local policy routing.

8. **next-key**—Set this parameter to $TO, $FROM, or $PAI if you wish to override the recently-returned lookup key value for the next stage.

9. Save and activate your configuration.

Maintenance and Troubleshooting

The **show sipd policy** command includes four additional counters that refer to single and multistage local policy lookups. All counters are reported for the recent period, and lifetime total and lifetime period maximum. These counters are:

- Local Policy Inits—Number of times the Oracle USM makes an initial local policy lookup.
- Local Policy Results Max—Number of times the Oracle USM truncated the number of routes returned for a local policy lookup because the maximum number of routes per local policy lookup (**max lp lookups routes per lookup**) threshold was reached.
- Local Policy Exceeded—Number of times the Oracle USM truncated the number of routes returned for a local policy lookup because the maximum number of routes per message request (**total lp routes**) threshold was reached.

- Local Policy Loops—Number of times the Oracle USM detected a loop while performing a multistage local policy lookup.

Traps

An SNMP trap is generated to notify that the limit on the **additional lp lookups** threshold has been reached during the recent window period. This trap occurs a maximum of once during a window period.

```
apSysMgmtLPLookupExceededTrap NOTIFICATION-TYPE
    STATUS          current
       DESCRIPTION
          " The trap will be generated the first time the additional Local
Policy Lookups limit is reached is in the recent window period. The trap will
only occur once during a window period."
       ::= { apSystemManagementMonitors 65}
```

Routing Based on UA Capabilities

In compliance with RFC 3841, the Oracle USM is able to make forwarding and forking decisions based on preferences indicated by the UA. To do this, the Oracle USM evaluates each callee's AOR contact to determine the capabilities advertised by the UA and uses this information to make forwarding and forking decisions.

Prior to this support, the Oracle USM made routing preference decisions solely via the q value present in the contact header. In cases where the preferences were equal, the Oracle USM simply forwarded to those contacts simultaneously (parallel forking). In cases where the q value were not equal, the Oracle USM forwarded in sequence (sequential forking), forwarding to the highest q value first.

The Oracle USM now extends upon this functionality by scoring contacts, based on their capabilities, and making forwarding decisions using that score in addition to the q value.

There is no additional Oracle USM configuration required to enable or invoke this processing. This functionality is supported for HSS, ENUM and Local Database configurations.

UE Capabilities

RFC2533 includes a framework that defines feature sets. Feature sets make up a group of media capabilities supported by a UA, individually referred to as media feature tags. In session networks, feature tag information is converted to a form specified in RFC3840 and exchanged between devices in the network to establish lists of UA capabilities. Based on these capabilities, session operation procedures are performed that facilitate preferred communications modalities.

RFC3840 defines:

- The format a UA uses to specify feature sets
- How a UA registers capabilities within the network
- An extension to the contact header so that it can include feature parameters
- The media tags that specify each capability

The full list of applicable media tags is presented in RFC 3840. Examples of tags include audio, automata, data, mobility, application and video.

Registering Capabilities at the Oracle USM

Endpoints register their capabilities by listing them in the Contact headers of the REGISTER request. The Oracle USM stores these feature parameters in its registration cache along with the other contact information. In the case of ENUM databases, the Oracle USM also sends capabilities information to the ENUM infrastructure so that it can maintain capabilities records.

In addition to the standard set of tags, the Oracle USM supports storing custom feature tags. Tags formatted with a + sign preceding the tag are recognized as custom tags. The exception to this are tags formatted using +sip.<tagname>, which are registered sip media feature tags.

An example of a contact header specifying audio, video and methods capabilities is shown below:

```
Contact: sip:u1@h.example.com;audio;video;methods="INVITE,BYE";q=0.2
```

Preferential Routing

The Oracle USM routes using UA capabilities only when acting as S-CSCF. It calculates preferred forwarding and forking using this information in conjunction with UA requests. This calculation is based on Preferential Routing, as defined in RFC3841. Note that the q value is used in this calculation.

Using Preferential Routing, the Oracle USM creates a target UA list from applicable contacts by matching capabilities with preferences. After creating the match list, it scores UEs based on how closely they match the preferred criteria. The system determines the forwarding order referring to the q value first and then the routing score. UEs for which both scores are equal are forwarded at the same time. All remaining UEs are forwarded sequentially.

The table below presents an example wherein the result of matching and scoring calculations causes the Oracle USM to forwards sequentially to UE3, then UE2, then UE1.

User Agent	q Value	Preferential Score
UE3	1000	1000
UE1	500	1000
UE2	1000	700

UAs may or may not include capability request information in their messages. Preferential routing processing accounts for this by defining both explicit and implicit feature preference processing procedures.

Explicit Feature Preference

RFC3841 defines the two headers that a UA can use to explicitly specify the features the target UA must support, including:

`Accept-Contact:` — UEs the session initiator would like to reach

`Reject-Contact:` — UEs the session initiator does not want to reach

When the Oracle USM receives messages that includes these headers, it gathers all the contacts associated with the AOR and creates a target list using preferential routing calculations. The example below, drawn from RFC 3841, specifies the desire to route to a mobile device that can accept the INVITE method.

```
Accept-Contact: *;mobility="mobile";methods="INVITE"
```

The “require” and explicit Feature Tag Parameters

RFC 3841 defines operational procedures based on the require and explicit feature tag parameters, which the Oracle USM fully supports. UAs include these parameters in the accept-contact: header to further clarify capabilities requirements for the session. The Oracle USM can use these parameters to exclude contacts or specify the forwarding order.

To summarize the use of these parameters per RFC 3841:

When both parameters are present, the Oracle USM only forwards to contacts that support the features and have registered that support.

If only the require parameter is present, the Oracle USM includes all contacts in the contact list, but uses a forwarding sequence that places the “best” match (with the most matching capabilities) first from those with the same q value.

If only the explicit parameter is present, the Oracle USM includes all contacts in the contact list, but uses a forwarding sequence that places contacts that have explicitly indicated matching capabilities before those with the same q value. Unlike requests that specify both require and explicit, non-matching contacts may be tried if the matching ones fail.

If neither parameter is present, the Oracle USM includes all contacts in the contact list, but determines a “best” match based on the “closest” match to the desired capabilities. Again the forwarding order starts with contacts that have the same q value.

Note that this preferential routing sequence can proceed with attempts to reach contacts with a lower q value after the sequences above are exhausted. Note also that the orders calculated by preferential routing never override any forwarding order specified by the UA.

Implicit Feature Preference

If the caller does not include accept-contact or reject-contacts in the message, the Oracle USM makes implicit feature preference assumptions. Implicit feature preference forwards messages to target UEs that support the applicable method, and, in the case of SUBSCRIBE requests, that support the applicable event.

For implicit feature preference cases, the Oracle USM uses the UE’s q value solely to determine parallel and sequential forking.

Supporting Media Sessions Established via ICE

The Oracle USM supports media session establishment for stations running Interactive Connectivity Establishment (ICE) despite the addressing changes made during normal operations. By inserting Oracle USM media interface addresses as relay candidates for all components (RTP, RTCP, and so forth) within the SDP offer/answer, the Oracle USM prevents ICE mismatch detection that would otherwise stop further ICE processing, including pair nomination and status updating. If the Oracle USM did not do this, the call would then proceed under the control of the B2BUA, which would select the media path instead of ICE.

As defined in RFC 5245, ICE defines the insertion of "candidate" addressing within the SDP offer/answer messaging. In conjunction with STUN (RFC 5389), the associated call setup procedure includes connectivity checks to these addresses, which must pass before a media session can proceed. This mechanism helps overcome a variety of network intricacies, including NAT traversal. In addition, RFC 5245 specifies a security mechanism that requires

the address outlined in the default media destination be included as a candidate. This default media destination is presented in the SDP's "m" and "c" lines. Normal SBC operations, to which the Oracle USM complies, includes altering this data to support media negotiation and/or media transport. This scenario causes an ICE mismatch. This feature prevents these mismatches.

Users configure this support on the Oracle USM using a **sip-config** option called **turn-on-the-fly**. With this option enabled, the Oracle USM inserts its own addressing as a relay candidate. This mechanism is based on RFC 5766 Traversal Using Relays around NAT (TURN), which itself is an extension of RFC 5389, Session Traversal Utilities for NAT (STUN).

The Oracle USM inserts itself as a lowest priority candidate. This ensures that ICE would choose the Oracle USM as a participant in the media path last.

In the SDP example below, the Oracle USM has replaced address and port information in the c and m lines with its own media port information. In this case, ICE checks determine that the Oracle USM address is not in the candidate list, to the media flows normally without further ICE processing.

```
c=IN IP4 172.16.101.15
m=audio 15086 RTP/AVP 103

a=candidate:5c0a80165 1 UDP
2130706431 172.16.100.166 2340 typ srflx
raddr 192.168.1.101 rport 2340
a=candidate:Hc0a80165 1 UDP
1694498815 192.168.1.101 2340 typ host
```

In the SDP example below, the user has enabled the **turn-on-the-fly** option, so the Oracle USM inserts itself as a relay candidate. The ICE checks pass and the media proceeds using ICE processing. Note also the priority designation of 255, which is ICE's lowest priority.

```
c=IN IP4 172.16.101.15
m=audio 15086 RTP/AVP 103

a=candidate:5c0a80165 1 UDP
2130706431 172.16.100.166 2340 typ srflx
raddr 192.168.1.101 rport 2340
a=candidate:Hc0a80165 1 UDP
1694498815 192.168.1.101 2340 typ host
a=candidate:C214c2Jj 1 UDP 255
172.16.101.15 15086 typ relay raddr
172.16.101.15 rport 15086
```

Further implementation details include:

- The Oracle USM uses latching to specify media flow source and destination addressing for endpoints behind a NAT. This is also true for RTCP, requiring the user to enable the **hnt-rtcp** parameter in the **media-manager** element.
- To ensure that the IP address and port pair used by the applicable outbound flow is reused for the inbound flow, enable the **symmetric-latching** parameter in the access signaling **realm** element.
- The Oracle USM disables flow guard timing for calls whose media paths are identified by ICE. Normal SIP session timers take down calls that fail to terminate.
- The Oracle USM sets up NAT entries assuming the RTCP address is (RTP IP address:RTP port +1).

- Use media management on the Oracle USM in conjunction with this **turn-on-the-fly** configuration. If not, the resultant media release by the Oracle USM causes unpredictable media path selection.

Configuring Support for ICE

Use the two configurations, shown below, to enable the Oracle USM to properly manage SDP for ICE processing.

1. From superuser mode, use the following command sequence to access **sip-config** configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ```

2. Use the **select** command to access the **sip-config** element.
3. Enable the **turn-on-the-fly** option.

   ```
   ORACLE(sip-config)# +option turn-on-the-fly enabled
   ORACLE(sip-config)# done
   ORACLE(sip-config)# exit
   ```

4. Use the **hnt-rtcp** command to enable the insertion of addressing to accommodate NAT traversal by RTCP.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# hnt-rtcp enabled
   ```

5. Navigate to the access realm element to enable **symmetric latching**.

   ```
   ORACLE(media-manager)# done
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)# select
   ```

   Select the realm to which the source endpoint belongs.

   ```
   ORACLE(realm-config)# symmetric-latching enabled
   ```

6. Use **done**, exit configuration mode, and run **verify-config** to complete ICE support configuration.

   ```
   ORACLE(realm-config)# done
   ORACLE(media-manager)# done
   ORACLE(media-manager)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE#
   ```

Routing-based RN and CIC

When the Oracle USM performs local policy routing, it selects local policy entries based on from addresses, to addresses, and source realms. All three are configurable in the local policy configuration. The to addresses can either be the username in a Request-URI (if it is an E.164/ phone number format), or the request-URI's hostname or IP address. The Oracle USM sorts matching local policies based on policy attribute entries. A policy attribute defines a next hop, which can be a session agent or a session agent group. Alternatively, the next hop might define an ENUM server group or local route table to use to find the next hop.

If the routing-based RN and CIC feature is not enabled, the Oracle USM performs the subsequent ENUM query or local route table lookup using the Request-URI's username, if it is a telephone number (TN). The TN is the normalized user part of the Request-URI, ignoring any user parameters or non-digit characters.

If the routing-based RN and CIC feature is enabled, the Oracle USM instead performs the ENUM or local route table lookup based on a user parameter, which is useful for lookups based on routing number (RN) or carrier identification code (CIC):

- An RN is a number that identifies terminating switch nodes in Number Portability scenarios when the original TN has been moved to the switch defined by the RN.
- A CIC is the globally unique number of the terminating carrier to which a ported number has been moved.

In applications where the Oracle USM is given the RN or the CIC in the Request-URI, this feature is useful because the Oracle USM can perform an additional ENUM or local route table lookup to find the next hop to the RN or the CIC. Typically, ENUM servers have imported Number Portability data with which to respond to the Oracle USM query, and (for example) the Oracle USM can use local route tables for storing CIC values for direct carrier hand-off.

Even with this feature enabled, the Oracle USM still performs local policy match selection based on the TN. This feature only uses the RN or CIC user-parameter for the ENUM or local route table lookup after the local policy and policy attributes have been selected.

Routing-based RN Configuration

This section shows you how to specify that a set of local policy attributes should use an RN for lookup. You can also set this value to CIC, or to any value you require.

You can set the lookup key to an RN in the local policy attributes' **next-hop** parameter.

To set the lookup key to RN:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-policy** and press Enter.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **policy-attributes** and press Enter.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

5. **next-hop**—In the **next-hop** parameter—after the kind of ENUM service used—type a colon (;). Then, without spaces, type in **key=rn** and press Enter.

   ```
   ORACLE(local-policy-attributes)# next-hop lrt:lookup;key=rn
   ```

6. Save and activate your configuration.

Codec Policies for SIP

The Oracle USM has the ability to add, strip, and reorder codecs for SIP sessions. This builds on the Oracle USM's pre-existing abilities to route by codec and reorder one codec in an SDP offer by allowing you to configure the order of multiple codecs and to remove specific codecs within the media descriptions in SDP offers.

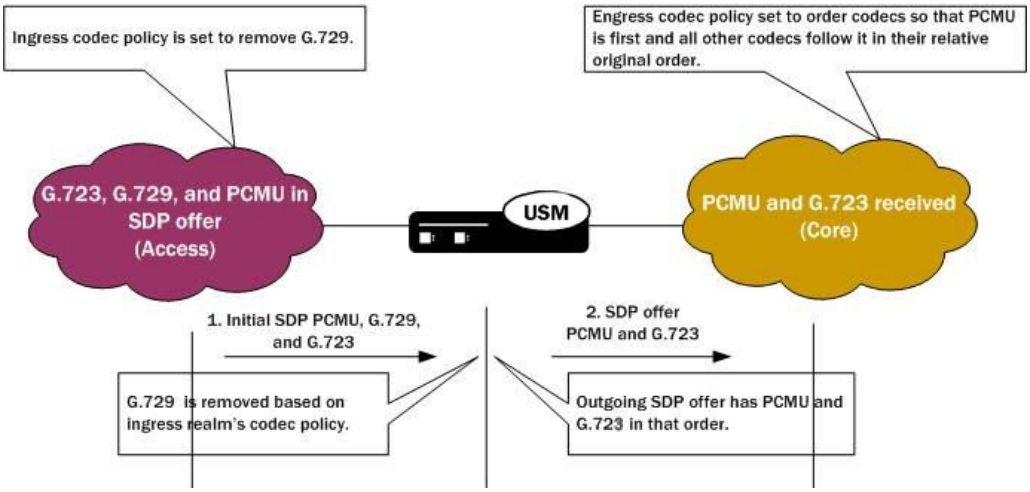
You can enable the Oracle USM to perform these operations on SDP offers by configuring codec policies. Codec policies are sets of rules that specify the manipulations to be performed on SDP offers. They are applied on an ingress and egress basis using the realm and session agent configurations.

Oracle USM supports three types of codec policies:

- Ingress policy—Codec policy that the Oracle USM applies to the SDP offer for incoming traffic
- Egress policy—Codec policy that the Oracle USM applies to the SDP offer for traffic leaving the Oracle USM
- Conditional policy—Codec policy that the Oracle USM applies to the SDP offer for traffic leaving the Oracle USM. A conditional policy differs from an egress policy in providing the capability to perform standard codec manipulations (add and strip) dynamically, based on the codec list and associated parameters contained in the original SDP offer.

The Oracle USM applies codec policies during the offer phase of media format negotiation. If codec manipulation is enabled, then the Oracle USM performs the modification according to the specific policy and forwards on the traffic.

For example, when the Oracle USM receives a SIP INVITE with SDP, it refers to the realm through which the INVITE arrived and performs any manipulations specified by an ingress codec policy that may have been assigned to the ingress realm. With the media description possibly changed according to the ingress codec policy, the Oracle USM passes the SDP offer to the outgoing realm so that the an egress codec policy can be applied. Note that the SDP to be evaluated by the egress codec policy may match the original SDP, or it may have been changed during transit through the ingress realm. After applying the egress coded policy, the Oracle USM forwards the INVITE.



Since the offer-answer exchange can occur at different stages of SIP messaging, the assigned ingress and egress roles follow the media direction rather than the signaling direction. It might be, for example, that the offer is in an OK that the Oracle USM modifies.

You can apply codec policies to realms and to session agents; codec policies configured in session agents take precedence over those applied to realms. However, it is not required that there be both an ingress and an egress policy either for realms or for session agents. If either one is unspecified, then no modifications take place on that side. If neither ingress nor egress policies specified, SDP offers are forwarded as received.

Relationship to Media Profiles

For each codec that you specify in a codec policy, there must be a corresponding media profile configuration on the Oracle USM. You configure media profiles in the ACLI via the session-router path. In them, you can specify codec type, transport protocol, required bandwidth, and a number of constraints.

Manipulation Modes

You can configure a codec policy to perform several different kinds of manipulations:

- Allow—List of codecs that are allowed for a certain codec policy; if a codec does not appear on this list, then the Oracle USM removes it. You can wildcard this list with an asterisk (*) so that all codecs are allowed. Further, you can create exceptions to a wildcarded allow list.
  - You make an exception to the wildcarded list of codecs by entering the codec(s) that are not allowed with a **no** attribute. This tells the Oracle USM to allow all codecs except the one(s) you specify.

    ```
    ACMEPACKET(codec-policy)# allow-codecs (* PCMA:no)
    ```

  - You can also create exceptions to allow lists such that audio or video codecs are removed. However, when the allow list specifies the removal of all audio codecs and an INVITE arrives at the Oracle USM with only audio codecs, the Oracle USM behaves in accordance with RFC 3264. This means that the resulting SDP will contain one attribute line, with the media port for the media line set to 0. The terminating side will need to supply new SDP in its reply because the result of the manipulation is the same as an INVITE with no body.

    ```
    ACMEPACKET(codec-policy)# allow-codecs (* audio:no)
    ```

- Order—List of the codecs where you specify their preferred order in the outgoing media offer. The Oracle USM arranges matching codecs according to the rule you set, and any remaining ones are added to the list in the same relative order they took in the incoming media offer. If your list specifies a codec that is not present, then the ordering proceeds as specified but skips the missing codec.
  You can use an asterisk (*) as a wildcard in this list, too. The placement of the asterisk is key, as you can see in the following examples:
  - For an order rule set this way

    ```
    ACMEPACKET(codec-policy)# order (A B C *)
    ```

    codecs A, B, and C will be placed at the front of the codec list in the order specified; all other codecs in the offer will follow A, B, and C in the same relative order they had in the original SDP offer.
  - For an order rule set this way:

    ```
    ACMEPACKET(codec-policy)# order (* A B C)
    ```

codecs A, B, and C will be placed at the end of the codec list in the order specified; all other codecs in the offer will come before A, B, and C in the same relative order they had in the original SDP offer.

  - For an order rule set this way

    ```
    ACMEPACKET(codec-policy)# order (A * B C)
    ```

    codec A will be placed at the beginning of the codec list, to be followed by all other codecs in the offer in the same relative order they had in the original SDP offer, and then B and C will end the list.

- Force—An attribute you can use in the allow list with one codec to specify that all other codecs should be stripped from the outgoing offer. You can specify multiple forced codecs in your rules.
  - If you set multiple codecs in the allow list and one of them is forced, then the outgoing offer will contain the forced codec.
  - If you set multiple codecs in the allow list and the one that is forced is not present in the offer, then the Oracle USM will select a non-forced codec for the outgoing offer.

    ```
    ACMEPACKET(codec-policy)# allow (PCMU G729:force)
    ```

  You cannot use the force attribute with a wildcarded allow list.
- No—An attribute that allows you to strip specified codecs or codec types from a wildcarded allow list.

  ```
  ACMEPACKET(codec-policy)# allow (* PCMA:no)
  ```

In-Realm Codec Manipulation

In addition to being able to apply codec policies in realms, the realm configuration supports a setting for determining whether codec manipulation should be applied to sessions between endpoints in the same realm.

In-realm codec manipulation can be used for simple call flows that traverse two realms. If the originating and terminating realms are the same, the Oracle USM checks to see if you have enabled this capability. If you have enabled it, then the Oracle USM performs the specified manipulations. If this capability is not enabled, or if the realm's media management in realm (**mm-in-realm**) setting is disabled, then the Oracle USM does not perform codec manipulations.

For more complex calls scenarios that involve call agent or reinitiation of a call back to the same realm, the Oracle USM does not perform in-realm codec manipulation.

Codec Policy Configuration

This section gives instructions and examples for how to configure codec policies and then apply them to realms and session agents. It also shows you how to configure settings for in-realm codec manipulation.

Creating a Codec Policy

To create a codec policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

```
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)#
```

3. Type codec-policy and then press Enter.

```
ACMEPACKET(media-manager)# codec-policy
ACMEPACKET(codec-policy)#
```

4. **name**—Enter the unique name for the codec policy. This is the value you will use to refer to this codec policy when you apply it to realms or session agents. This parameter is required and is empty by default.
5. **allow-codecs**—Enter the list of media format types (codecs) to allow for this codec policy. In your entries, you can use the asterisk (*) as a wildcard, the force attribute, or the no attribute so that the allow list you enter directly reflects your configuration needs. Enclose entries of multiple values in parentheses (( )).

   The codecs that you enter here must have corresponding media profile configurations.
6. **add-codecs-on-egress**—Enter the codecs that the Oracle USM adds to an egress SDP offer if that codec is not already there. This parameter applies only to egress offers.

   The codecs that you enter here must have corresponding media profile configurations.

   add-codecs-on-egress can be used to construct ingress, egress, or conditional codec policies.
7. **order-codecs**—Enter the order in which you want codecs to appear in the outgoing SDP offer. Remember that you can use the asterisk (*) as a wildcard in different positions of the order to directly reflect your configuration needs. Enclose entries of multiple values in parentheses (( )).

   The codecs that you enter here must have corresponding media profile configurations.
8. Save and activate your configuration.

   Your codec policy configuration will resemble the following example:

```
codec-policy
        name            private
        allow-codecs    g723:no pcmu video:no
        order-codecs    pcmu
```

Applying a Codec Policy to a Realm

Note that codec policies defined for session agents always take precedence over those defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

Applying a Codec Policy to a Session Agent

Note that codec policies that are defined for session agents always take precedence over those that are defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

In-Realm Codec Manipulations

To enable in-realm codec manipulations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-manip-in-realm**—Enter the name of the codec policy that you want to apply to this realm. The default value is **disabled**. The valid values are:

   - enabled | disabled

5. Save and activate your configuration.

QoS Based Routing

In addition to configuring your system for routing based on certain session constraints, you can also set up routing based on QoS. QoS based routing uses the R-Factor on a per-realm basis to either cut back on the traffic allowed by a specific realm, or to shut that traffic off altogether.

To use this feature, you set up QoS constraints configurations and apply one per realm. The QoS constraints configuration allows you to set up two thresholds:

- Major—The major threshold sets the R-Factor limit beyond which the Oracle USM rejects a certain percentage (that you configure) of calls. That is to say, it rejects inbound calls at the rate you set with a 503 Service Unavailable status code, and rejects outbound calls if there are no alternative routes.
- Critical—The critical threshold, when exceeded, causes the Oracle USM to behave the same way it does when any of the session constraints (set in the session-constraints configuration) are exceeded. All inbound calls to the realm are rejected with a 503 Service Unavailable status code, and (if there is no alternate route) outbound calls are rejected, too. Until the R-Factor falls within acceptable means and the session constraint's time-to-resume value has elapsed, the realm remains in this state.

Management

This feature is supported by MIBs and traps. Historical data recording (HDR) also supports this feature by providing the following metrics in the session realm statistics collection group:

- Average QoS RFactor (0-93)
- Maximum QoS RFactor (0-93)
- Current QoS Major Exceeded
- Total QoS Major Exceeded
- Current QoS Critical Exceeded
- Total QoS Critical Exceeded

QoS Contraints Configuration

This section shows you how to configure a QoS constraints configuration and then how to apply it to a realm.

Configuring QoS Constraints

Your first step to enabling QoS based routing is to set up a QoS constraints configuration. This configuration is where you enter major and critical thresholds, as well as the load reduction for the realm should the R-Factor exceed the major threshold.

To set up a QoS constraints configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **qos-constraints** and press Enter.

   ```
   ORACLE(session-router)# qos-constraints
   ORACLE(qos-constraints)#
   ```

4. **name**—Enter the name of this QoS constraints configuration. This parameter uniquely identifies the configuration, and you use this value when applying the configuration to a realm. This parameter has no default and is required.
5. **state**—Set the state of this QoS constraints configuration. The default is **enabled**, but you can set this parameter to **disabled** if you want to stop applying these constraints.
6. **major-rfactor**—Enter a numeric value between **0** (default) and **9321** to set the threshold that determines when the Oracle USM applies the call reduction rate. If you leave this parameter set to **0**, then the Oracle USM will not apply a major threshold for any realm where you apply this QoS constraints configuration.

   Note that this value must be greater than that you set for the **critical-rfactor**, except when the **major-rfactor** is 0.
7. **critical-rfactor**—Enter a numeric value between **0** (default) and **9321** to set the threshold that determines when the Oracle USM rejects all inbound calls for the realm, and rejects outbound calls when there is no alternate route. If you leave this parameter set to **0**, then the Oracle USM will not apply a critical threshold for any realm where you apply this QoS constraints configuration.

   Note that this value must be less than that you set for the **major-rfactor**, except when the **major-rfactor** is 0.
8. **call-load-reduction**—Enter a number from **0** (default) to **100** representing the percentage by which the Oracle USM will reduce calls to the realm if the **major-rfactor** is exceeded. If you leave this parameter set to 0, then the Oracle USM will not reduce call load for the realm—even when the major-rfactor is configured.

   This is the percentage of inbound and outbound calls the Oracle USM will reject. For example, if you set this parameter to 50 and the major threshold is exceeded, then the Oracle USM rejects every other call to the realm.
9. Save and activate your configuration.

Applying QoS Constraint to a Realm

You apply QoS constraints to realms using the **qos-constraint** parameter.

To apply a QoS constraint to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```
2. Type **media-manager** and press Enter

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```
3. Type **realm-config** and press Enter. If you adding this feature to a pre-existing realm, then you need to select and edit that realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```
4. **qos-constraints**—Enter the name value from the QoS constraints configuration you want to apply to this realm.

   Save and activate your configuration.

7

ENUM Based Oracle USM

The Oracle USM contacts an ENUM server via DNS for two purposes:

- to obtain authentication data for a registering UA (often referred to as a UE)
- to query the user subscriber database (ENUM) regarding the registration state of the AoR and update the database with the latest information

Message Authentication for SIP Requests

The Oracle USM authenticates requests by configuring the sip authentication profile configuration element. The name of this configuration element is either configured as a parameter in the sip registrar configuration element's authentication profile parameter or in the sip interface configuration element's sip-authentication-profile parameter. This means that the Oracle USM can perform SIP digest authentication either globally, per domain of the Request URI or as received on a SIP interface.

After naming a sip authentication profile, the received methods that trigger digest authentication are configured in the methods parameter. You can also define which anonymous endpoints are subject to authentication based on the request method they send to the Oracle USM by configuring in the anonymous-methods parameter. Consider the following three scenarios:

1. By configuring the methods parameter with REGISTER and leaving the anonymous-methods parameter blank, the Oracle USM authenticates only REGISTER request messages, all other requests are unauthenticated.
2. By configuring the methods parameter with REGISTER and INVITE, and leaving the anonymous-methods parameter blank, the Oracle USM authenticates all REGISTER and INVITE request messages from both registered and anonymous endpoints, all other requests are unauthenticated.
3. By configuring the methods parameter with REGISTER and configuring the anonymous-methods parameter with INVITE, the Oracle USM authenticates REGISTER request messages from all endpoints, while INVITES are only authenticated from anonymous endpoints.

Credential Retrieval

The Oracle USM requests authentication information from an ENUM server via DNS when it receives a REGISTER or other message from an endpoint. This server, which provides authentication information, is defined on the Oracle USM in an enum-config configuration element that includes an enum-servers (the IP addresses of the servers) and realm parameters. Together, these two parameters define the DNS/ENUM server(s) which provide authentication data.

The target ENUM server is determined first by setting the credential retrieval config parameter to **enum-config** so the Oracle USM will reference that enum config. Next, set the credential retrieval config parameter to the name of an enum config configuration element which is populated with the ENUM servers' IP addresses.

User Authentication Query

As soon as a request is received on a SIP interface and has been determined to require authentication, the Oracle USM attempts to authenticate the endpoint. It sends a DNS TXT query including the UA's AoR to an ENUM database and expects the H(A1) defined in RFC2617 for the user being authenticated.

SIP Digest User Authentication

SIP Authentication Challenge

When the Oracle USM receives a response from the ENUM server including the hash value for the user, it sends a SIP authentication challenge to the endpoint, if the endpoint did not provide any authentication headers in its initial contact the with Oracle USM. If the endpoint is registering, the Oracle USM replies with a 401 Unauthorized message with the following WWW-Authenticate header:

```
WWW-Authenticate: Digest realm="atlanta.com", domain="sip:boxesbybob.com",
qop="auth", nonce="f84f1cec41e6cbe5aea9c8e88d359", opaque="", stale=FALSE,
algorithm=MD5
```

If the endpoint initiates any other request to the Oracle USM besides REGISTER, the Oracle USM replies with a 407 Proxy Authentication Required message with the following Proxy-Authenticate header:

```
Proxy-Authenticate: Digest realm="atlanta.com", qop="auth",
nonce="f84f1cec41e6cbe5aea9c8e88d359", opaque="", stale=FALSE, algorithm=MD5
```

Authentication Header Elements

- Digest Realm—This value is configured in the digest-realm parameter in the sip-registrar configuration element. This parameter is mandatory when using the "ENUM-TXT" credential retrieval method.
- Domain—A quoted, space-separated list of URIs that defines the protection space. This is an optional parameter for the "WWW-Authenticate" header.
- Nonce—A unique string generated each time a 401/407 response is sent.
- Qop—A mandatory parameter that is populated with a value of "auth" indicating authentication.
- Opaque—A string of data, specified by the Oracle USM which should be returned by the client unchanged in the Authorization header of subsequent requests with URIs in the same protection space.
- Stale—A flag indicating that the previous request from the client was rejected because the nonce value was stale. This is set to true by the SD when it receives an invalid nonce but a valid digest for that nonce.
- Algorithm—The Oracle USM always sends a value of "MD5"

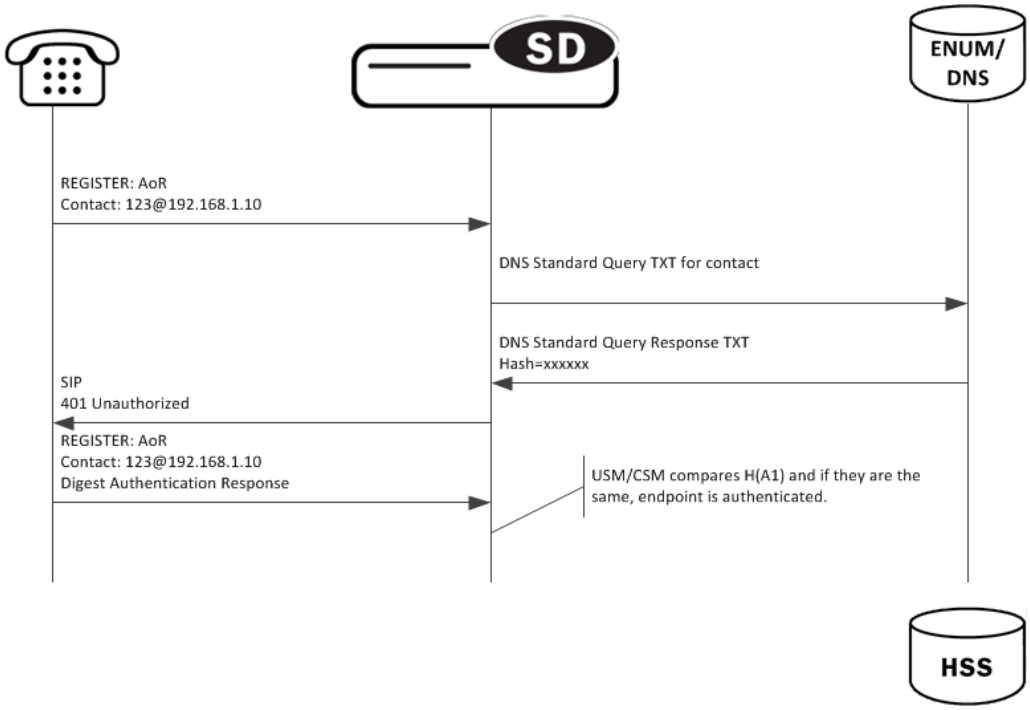
SIP Authentication Response

After receiving the 401/407 message from the Oracle USM, the UA resubmits its original request with an Authorization: header including its own internally generated MD5 hash.

Oracle USM Authentication Check

At this point, the Oracle USM has received an MD5 hash from the ENUM server and an MD5 hash from the UA. The Oracle USM compares the two values and if they are identical, the endpoint is successfully authenticated. Failure to match the two hash values results in a 403 or 503 sent to the authenticating endpoint.

The following image shows the user authentication process.



Oracle USM as Registrar

DDNS Update to User Subscriber Database

As REGISTER messages are received, the Oracle USM updates the ENUM database via DDNS UPDATE messages as defined by RFC2136. New registrations are added to the database while expired or deleted registrations are removed.

The Oracle USM acts as a registrar by configuring the sip registrar configuration element. When registrar functionality is enabled, the Oracle USM acts as a registrar rather than only caching and forwarding registrations to another device. Oracle USM registry services are enabled globally per domain, not on individual SIP interfaces or other remote logical entities.

On receiving a REGISTER message, the Oracle USM checks if it is responsible for the domain contained in the Request-URI as defined by the domains parameter and finds the corresponding sip registrar configuration. This is a global parameter and all messages are checked against all sip registrar domains. Thus you could create one sip registrar configuration element to handle all *.com domains and one sip registrar configuration element to handle all *.org domains. The Oracle USM begins registrar functions for all requests that match the configured domain per sip-registrar configuration element.

A UA is considered registered after the Oracle USM updates the ENUM server with a DDNS dynamic update. After this action, the Oracle USM sends a 200 OK message back to the registering UA.

TTL

Part of the Oracle USM architecture includes a local ENUM cache which maintains the results from ENUM queries locally. To enable Oracle USM, the TTL value in the enum config must be set to 0. This ensures that whenever an ENUM query is required, the Oracle USM consults the central User Subscriber Database to have the latest network-wide information about the UA it is trying to reach, instead of its ENUM cache.

ENUM Database Correlation

When a UA registers, as the number of associated contacts for an AoR grows or shrinks, the ENUM-based User Subscriber Database is updated in turn with the latest information using a DDNS UPDATE. After a REGISTER including authentication information is received from a UA, the Oracle USM sends a Standard NAPTR query for the AoR to the ENUM server. The ENUM server replies with a Standard Query response, including the NAPTR records.

After receiving the entries from the ENUM database via the ENUM query, the list must be correlated with the Oracle USM's view of the AoR's registration state. The differences will be resolved by sending a DDNS UPDATE to add or remove entries from the ENUM server. The database correlation phase only occurs when endpoints register.

Contacts that need to be added are put on the UPDATE add list. Contacts that are being unregistered (contact with Expires=0) or have expired timestamp are added to the UPDATE remove list.

Entry Expiration

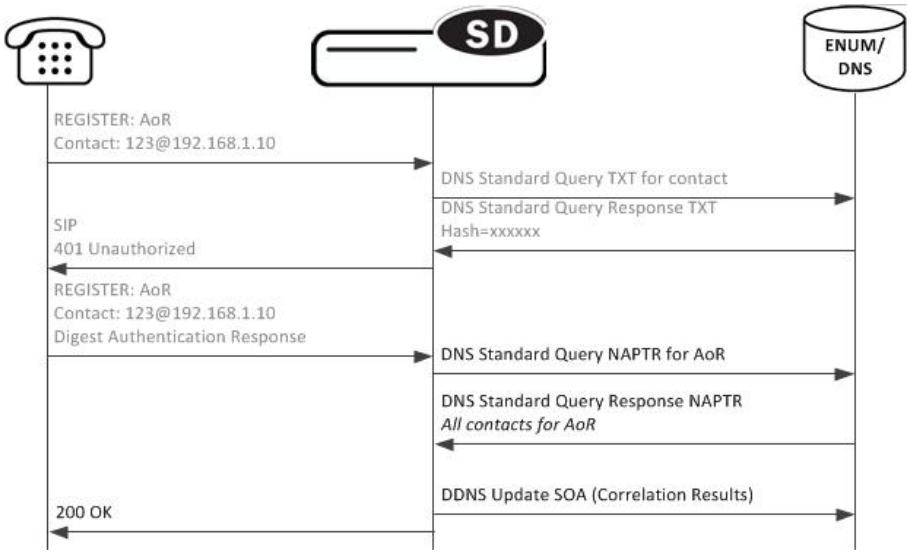
The Oracle USM employs a process to remove expired contacts from the ENUM database whether they were entered by the active Oracle USM or other Oracle USM. After determining that a contact is expired, the Oracle USM removes the contact in a subsequent DDNS update.

To do this, the Oracle USM includes an expiration parameter in the Contacts it insert into the ENUM database. Expiration is indicated with a ts= parameter. This parameter's value is set to the initial registration time measured on the Oracle USM plus the REGISTER message's Expires: header value or Expires parameter in the Contact header value. This value is measured in seconds after the epoch. In a DDNS update, a ts= parameter appears as follows:

```
Regex: "!^.*$!sip:234-hchse6c0d01u2@172.16.101.51:5060;ts=1313493824!"
```

After the Oracle USM retrieves contacts for an AoR in a NAPTR record that are expired, based on ts= parameters, the Oracle USM's next Dynamic update tells the server to remove those contacts from the ENUM database.

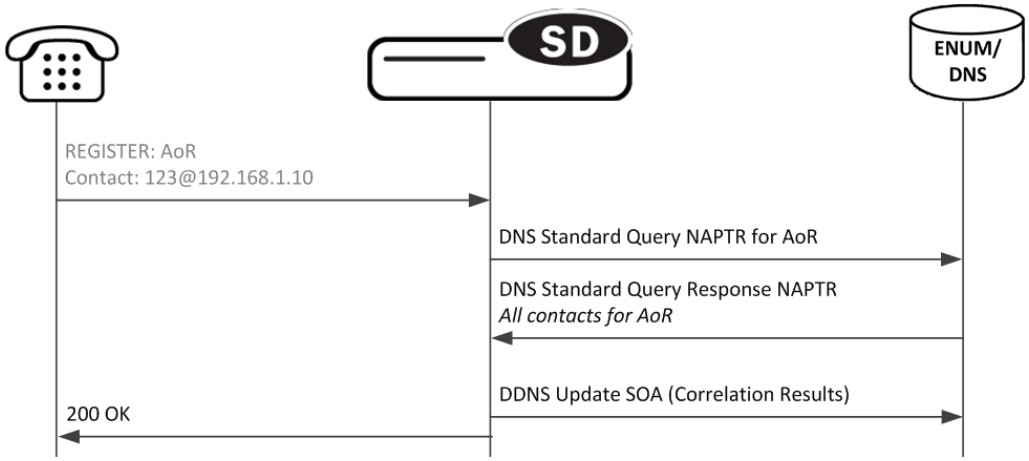
In this way, all Oracle USM members of a domain may remove expired contacts from the ENUM database.



Register Refresh

When a UA sends a register refresh, the Oracle USM first confirms that the authentication exists for that UA's registration cache entry, and then is valid for the REGISTER refresh. Then, the lifetime timer (value in Expires: header) for that registration cache entry is checked.

If the timer has not exceeded half of its lifetime, only a 200 OK is sent back to the UA. If the timer has exceeded half of its lifetime, the Oracle USM sends a NAPTR update to the ENUM database.



In addition to the baseline Oracle USM REGISTER refresh conditions, an ENUM database update is required when one of the following conditions is satisfied:

- The location update interval timer has expired—This value, configured in the sip registrar configuration element ensures that that ENUM database always has the latest user information by periodically sending Standard Queries.

- The message's call-id changes while the f**orward-reg-callid-change** option in the sip config configuration element is set. This covers the case where the UA changes the Oracle USM through which it attaches to the network.
- The REGISTER message's Cseq has skipped a number. This covers the case in which a user registered with Oracle USM1, moves to Oracle USM2, and then returns to Oracle USM1.
- The REGISTER message's contact list has changed.

After receiving the entries from the ENUM database via the NAPTR query, the list is correlated with the internal registration cache. Appropriate DDNS updates are preformed (see: ENUM Database Correlation).

If the Oracle USM updates the ENUM database due to one of the above conditions, the UA's access-side Expires timer is reset to the REGISTER message's Expires: header value, and returned in the 200 OK. This happens even in the case when the reREGISTER was received in the first half of the previous Expires period. In addition, the core-side location update interval timer is refreshed on both active and standby.

When the above four conditions are not met, the registration and reregistration expiration proceeds normally. If the access-side expiration timer has not exceeded half of its lifetime, only a 200 OK is sent back to the UA. If the timer has exceeded half of its lifetime, the Oracle USM refreshes the registration to the ENUM server. Note: Upon a Call-id or contact list change, both the registration cache timer and the ENUM database are updated.

Limiting AOR Contacts

The Oracle USM allows you to limit the number of contacts that apply to AORs. If the Oracle USM receives a registration request that exceeds the maximum that you configured, it responds with a local response, a 403 Forbidden by default, and does not register the additional contact. The system only rejects registration requests that exceed the maximum. Existing contacts persist normally.

The system checks against the maximum in the following circumstances:

- A new registration is received
- The location-update-interval expires
- A call-id changes (and the forward-reg-callid-change option is enabled)
- A registrar message sequence number has skipped a number
- There is any change to the contact list

If the number of contacts in the initial registration exceeds the maximum, the Oracle USM rejects the entire registration. In addition, if you configure this feature while the system is operational, your setting only applies to new registrations.

You configure these maximums on a per-registrar basis. The value ranges from 0-256. The feature is RTC supported.

User Registration based on Reg-ID and Instance-ID (RFC 5626)

Sometimes a user's device reregisters from a different network than its original registration. This event should be considered a location update rather that a completely new registration for the Contact. The Oracle USM can perform this way by considering the endpoint's reg-id and instance-id parameters defined in RFC 5626.

The Oracle USM identifies new REGISTER requests received on a different access network as a location update of the existing binding between the Contact and AoR. Without this feature, the Oracle USM would create a new binding and leave the old binding untouched in the local registration cache/ENUM database. This scenario is undesirable and leads to unnecessary load on various network elements including the Oracle USM itself.

The following conditions must be matched to equate a newly registering contact as a location update:

For a received REGISTER:

- The message must not have more than 1 Contact header while 1 of those Contact headers includes a reg-id parameter. (failure to pass this condition prompts the Oracle USM to reply to the requester with a 400 Bad Request).
- The Supported: header contains **outbound** value
- The Contact header contains a **reg-id** parameter
- The Contact header contains a **+sip.instance** parameter

After these steps are affirmed, the Oracle USM determines if it is the First hop. If there is only one Via: header in the REGISTER, the Oracle USM determines it is the first hop and continues to perform Outbound Registration Binding processing.

If there is more than 1 Via: header in the REGISTER message, the Oracle USM performs additional validation by checking that a Path: header corresponding to the last Via: includes an ob URI parameter, Outbound Registration Binding may continue.

If the Oracle USM is neither the first hop nor finds an ob URI in Path headers, it replies to the UA's REGISTER with a 439 First Hop Lack Outbound Support reply.

reREGISTER Example

The user (AoR) bob@example.com registers from a device +sip.instance= <urn:uuid:0001> with a reg-id ="1", contact URI = sip:1.1.1.1:5060. A binding is be created for bob@example.com+<urn:uuid:0001>+reg-id=1 at sip:1.1.1.1.:5060.

Next, Bob@example.com sends a reREGISTER with the same instance-id but with a different reg-id = 2 and contact URI = sip:2.2.2.2:5060.

The previous binding is removed. A binding for the new contact URI and reg-id is created. bob@example.com+<urn:uuid:0001>+reg-id=2 at sip:2.2.2.2:5060

Outbound Registration Binding Processing

An outbound registration binding is created between the AoR, instance-id, reg-id, Contact URI, and other contact parameters. This binding also stores the Path: header.

Matching re-registrations update the local registration cache as expected. REGISTER messages are replied to including a Require: header containing the outbound option-tag.

If the Oracle USM receives requests for the same AOR with some registrations with reg-id + instance-id and some without them, the Oracle USM will store them both as separate Contacts for the AOR; The AoR+sip.instance+reg-id combination becomes the key to this entry.

ENUM Database Update

When a REGISTER message is received:

1. The ENUM user database is queried for the AoR and any existing entries.
2. If there are any entries with the same instance-id as the current REGISTER request in the ENUM query response, then those entries will be marked for subsequent removal in the ENUM database.
3. The ENUM database is updated with a NAPTR request. This request adds the new Contact URI for that AOR+instance-id and removes any existing entries for the same AOR +instance-id.

NAPTR Update Format

The ENUM database update includes the instance-id and reg-id when those parameters are present in a registration. These values are appended to the regex replacement field. For example:

```
!^.*$!sip:3556-1cdstqjt90hve@172.16.101.62:5060;sip.instance=<urn:uuid:
00000000-0000-1000-8000-000A95A0E128>;reg-id=1;ts=1326568408;!
```

Oracle USM Licensing

The Oracle USM connected to an ENUM database requires two licenses: Registration Cache Limit, SIP Authorization/Authentication.

For ENUM-based Oracle USM, the SIP Authorization/Authentication license reveals the SIP Authentication Profile configuration element. Configuring both configuration elements is required to operate a Oracle USM. Refer to the Licensing and Database Registration Limits section for the third license required for Oracle USM operation.

Refer to the Oracle SBC ACLI Configuration guide, Getting Started chapter for how to install licenses in your system.

ACLI Instructions

ENUM Configuration

First the server used for authentication and as the User Subscriber Database is created.

To configure the ENUM Configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **enum-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)#
   ```

   You may now begin configuring the enum-config configuration element.

4. **name**—Set a name to use to reference this enum configuration from within the Oracle USM.
5. **top-level-domain**—Enter the domain which this ENUM server(s) services and returns results for.
6. **realm-id**—Enter the realm name where this ENUM server exists.
7. **enum-servers**—Enter the IP address of one or more ENUM servers used for registration. Multiple entries are separated by commas.
8. **service-type**—Leave this as its default.
9. **ttl**—Leave this at the default of 0 to set the TTL value (in seconds) for NAPTR entries as populated when sending a DNS update to the ENUM server.
10. **order**—Enter the value to populate the order field with when sending NAPTR entries to the ENUM server.
11. **preference**—Enter the value to populate the preference field with when sending NAPTR entries to the ENUM server.
12. Type **done** when finished.

SIP Authentication Profile

To configure the SIP Authentication Profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-authentication-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-authentication-profile
   ORACLE(sip-authentication-profile)#
   ```

   You may now begin configuring the SIP Authentication Profile configuration element.

4. **name**—Enter the name of this SIP authentication profile that will be referenced from a SIP registrar (or SIP interface).
5. **methods**—Enter all the methods that should be authenticated. Enclose multiple methods in quotes and separated by commas.
6. **anonymous-methods**—Enter the methods from anonymous users that require authentication. Enclose multiple methods in quotes and separated by commas.
7. **digest-realm**—enter the digest realm sent in an authentication challenge (401/407) sent to a UA. This is required in ENUM/DNS deployments,.

8. **credential-retrieval-method**—Enter **ENUM-TXT**.
9. **credential-retrieval-config**—Enter the enum-config name used for retrieving authentication data.
10. Type **done** when finished.

SIP Registrar

To configure the Oracle USM to act as a SIP Registrar:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)#
   ```

4. **name**—Enter a name for this SIP registrar configuration element.
5. **state**—Set this to **enabled** to use this SIP registrar configuration element.
6. **domains**—Enter one or more domains in the R-URI this configuration element handles. Wildcards are valid for this parameter. Multiple entries can be entered in quotes, separated by commas.
7. **subscriber-database-method**—Set this to **DDNS**.
8. **subscriber-database-config**—Enter the enum-config configuration element name that will handle REGISTER messages for this domain. This should be the same element used for requesting authentication data.
9. **authentication-profile**—Enter a sip-authentication-profile configuration element's name. The sip authentication profile object referenced here will be looked up for a REGISTER message with a matching domain in the request URI. You may also leave this blank for the receiving SIP Interface to handle which messages require authentication if so configured.
10. **location-update-interval**—Keep or change from the default of 1400 minutes (1 day).
11. Type **done** when finished.

Maximum Number of Contacts

To configure a sip-registrar with a maximum of 10 contacts per AOR:

1. From superuser mode, use the following command sequence to access sip-registrar element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)# select
   ```

   Select the registrar you want to configure.

2. Specify the number of contacts.

```
AORACLE(sip-registrar)# max-contacts-per-aor 10
AORACLE(sip-registrar)# done
```

Response to Exceeding Maximum Contacts

To configure local response for the Oracle USM to issue when max-contacts-per-aor is exceeded:

1. From superuser mode, use the following command sequence to access local-response and add an entry.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# local-response-map
   ```

2. Access the entries configuration.

   ```
   ORACLE(local-response-map)# entries
   ```

3. Specify the local error you need to configure.

   ```
   ORACLE(local-response-map-entry)# local-error contacts-per-aor-exceed
   ```

4. Specify the sip-reason for this error.

   ```
   ORACLE(local-response-map-entry)# sip-reason forbidden
   ```

5. Specify the error code for this error.

   ```
   ORACLE(local-response-map-entry)# sip-status 403
   ORACLE(local-response-map-entry)# done
   local-response-map-entry
           local-error                     contacts-per-aor-exceed
           sip-status                      403
           q850-cause                      0
           sip-reason                      forbidden
           q850-reason
           method
           register-response-expires
   ORACLE(local-response-map-entry)# exit
   ```

Update to ENUM Database on Endpoint Connection Loss

The Oracle USM can monitor an endpoint's transport-layer status for loss of connectivity in multiple ways. Then, when the endpoint's connection to the Oracle USM has been terminated, the Contact is removed from the registration cache, as associated under a registered AoR. In addition, the user database is immediately updated.

These transactions contribute to the registration cache and ENUM user database being updated in real time to retain only reachable contacts for a registered AoR. This feature helps to alleviate:

- unnecessary transactions and system load spent on attempting to reach an unreachable endpoint
- incorrect and out of date statistics

Connection Reuse

The connection between the Oracle USM and an endpoints must employ the connection reuse mechanism to enable this feature's de-registration and user database updates. Connection reuse is when an endpoint registers to the Oracle USM and all subsequent signaling between the Oracle USM and that endpoint reuses the same socket pair. There are four cases when connection reuse is enabled:

- Connection reuse is enabled on the SIP interface facing the endpoint(s). This is enabled by adding the **reuse-connections=yes** option on a SIP interface.
- The endpoint is behind a NAT. Because of the fundamental method that the Oracle USM uses for maintaining its connection to an endpoint behind a NAT, this case will always force connection reuse.
- The endpoint includes the alias parameter in the Via: header in its REGISTER message to the Oracle USM (RFC 5923).
- If the endpoint is configured as a session agent, the reuse-connections parameter must be set to TCP. When receiving signaling from a remote logical entity such as a session agent defined for an endpoint, if the reuse-connections parameter is set to tcp, the Oracle USM enables connection reuse between itself and the UA.

Unreachability Determination

There are four ways that the Oracle USM determines endpoint is not reachable:

- No CRLF message is returned to the Oracle USM within the expected time frame: this is based on the Oracle USM's RFC5626 support.
- No TCP Keepalive is returned to the Oracle USM within the expected timeframe. This is based on configuring the network parameter configuration element per application interface.
- The endpoint explicitly terminates its transport-layer connection to the Oracle USM.
- The endpoint is otherwise labeled as unreachable from a non-explicit fault condition for a call made to an unreachable endpoint.

RFC 5635 Failure

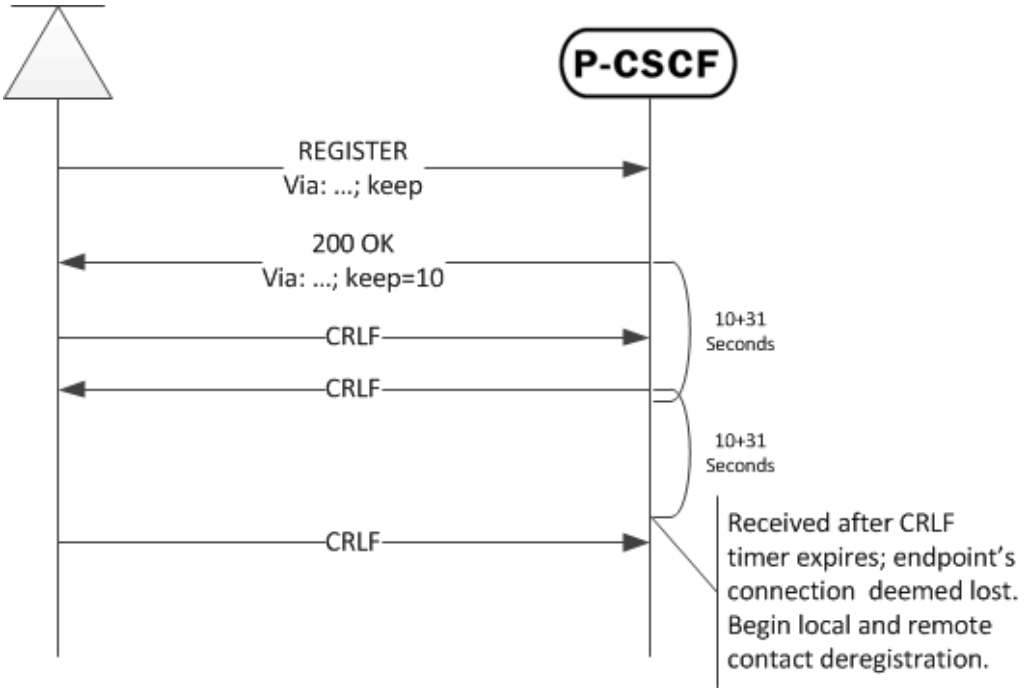
The Oracle USM supports the RFC 5635 method of SIP application keepalives, which are endpoint-initiated, i.e., the endpoint starts the mechanism by including the keep parameter in the initial Via: header. Endpoint reachability is determined the receipt or loss of a CR/LF ping-pong message. In the SIP interface configuration element, you set the **register keep alive** parameter to **always** or **bnat** (behind NAT), to enable RFC 5635 functionality. This applicable to TCP or TLS connections. **Always** forces the Oracle USM to always return a CRLF reply when the keep parameter is in the initial Via: header. **bnat** forces the Oracle USM to replies to RFC 5635 requests when the endpoint is located behind a NAT.

Next, you can accept the Oracle USM's default keep alive window of 30 seconds, or you may set your own by configuring the **tcp nat interval** or **inactive con timeout** value, both found in the sip interface configuration element. The Oracle USM uses the smaller of the two configured values. The chosen value is inserted into the keep parameter in the 200 OK message returned to the registering endpoint.

> **Note:**
>
> The inactive con timeout value otherwise disconnects a TCP/TLS connection after the configured value elapses. The tcp nat interval value is also inserted into the expires parameter in the Contact: header for devices identified as behind a NAT.

In addition, the Oracle USM maintains a keep timer. The Oracle USM adds 31 seconds to the keep value it returns and begins counting down. 31 is the chosen margin to account for any network or application delay.

If the endpoint returns the CRLF before the timer expires, the endpoint is considered up and a new CRLF ping is sent to the endpoint; the timer begins counting down again. If the Oracle USM fails to receive the CRLF ping before the timer expires, then the UA is considered unreachable. Provisions begin to remove that contact from the registration cache and then the ENUM user database.



TCP Keepalive Failure

Endpoint reachability can be determined from TCP keepalives, as enabled per SIP interface. This method of determining connectedness is based on configuring the global network parameters configuration element that is enabled on a SIP interface with the tcp-keepalive parameter. See the System TCP Keepalive Settings section of the Oracle SBC ACLI Configuration guide for how to configure the TCP keepalive feature. When an endpoint fails the TCP keepalive test, provisions begin to remove that contact from the registration cache and then the ENUM user database.

Explicit and undetermined connection termination

Ideally, a UA will gracefully close its TCP connection to the Oracle USM, and in turn the socket pair will be considered closed with the UA being unreachable. In less ideal cases, the UA goes dark and no response is received when expected. The Oracle USM considers the unresponsive UA as unreachable as call attempts time out. Provisions then begin to remove that contact from the registration cache and then the ENUM user database.

Registration Cache and User Database Removal

When the Oracle USM sets up the initial REGISTER request from a UA, an internal binding is created between the contact and the AoR for that user. If a UA is considered unreachable by either of the four conditions explained in the previous section, the following occur:

- The contact's entry in the registration cache is removed. If this is the last contact registered for an AoR, the entire entry for the AoR is removed from the registration cache.
- The Oracle USM sends an UPDATE to the ENUM user database removing the Contact.

To enable these actions, you must configure the **force unregistration** option sip config configuration element and also set the **unregister on connection loss** parameter in the sip interface configuration element to **enabled**.

ACLI Instructions

To globally enable force-unregistration at the sip-config level:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. Type **select** to continue.

   ```
   ORACLE(sip-config)# select
   ORACLE(sip-config)#
   ```

5. **options**—Set the options parameter by typing **options**, a Space, **force-unregistration** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-config)# options +force-unregistration
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Type **done** and **exit** to complete configuration of this **sip-config** configuration element.

SIP Interface Configuration

To configure the SIP Interface configuration element portion of the ENUM Database update feature:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and choose the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(sip-interface)# select
   <realm-id>:
   1: private 192.168.101.17:5060
   2: public  172.16.101.17:5060
   selection: 1
   ```

5. **unregister-on-connection-loss**—Set this parameter to **enabled** for the Oracle USM to update the ENUM server when an endpoint is deemed failed.

   Parameters for determining endpoint reachability:

6. **tcp-keepalive**—You may set this parameter to enabled to enforce the network-parameters configuration element located in the system-config path. See the Oracle SBC ACLI Configuration guide, System TCP Keepalive Settings section for more information.

7. **register-keep-alive**—Set this parameter to **always** for the Oracle USM to return the **keep** parameter, with optional value (as configured in the next two steps) in the Via: header to an endpoint including an empty keep value in its initial REGISTER message.

   You may set none, one, of both the following for a keep value returned to the initiating endpoint. Read the RFC 5635 Failure section for how the actual value is determined:

8. **inactive con timeout**—Set this parameter value to the value in seconds inserted in the returned keep parameter for RFC 5635 support

9. **tcp nat interval**—Set this parameter value to the value in seconds inserted in the returned keep parameter for RFC 5635 support

   Type **done** and **exit** to complete configuration of this **sip-interface** configuration element.

OAuth 2.0 Support

The Oracle USM supports Open Authorization (OAuth) in addition to SIP digest authentication for user authorization within ENUM deployments. Both authorization methods can be operational simultaneously, allowing some users to authorize via OAuth and others via SIP digest. Applicable scenarios include authorizing registrations, subscriptions and invites.

OAuth uses HTTP to provide end users with access to services from OAuth 2.0 protected resources using various clients. OAuth also allows users to authorize third-party access to their services using user-agent redirections rather than sharing username password pairs. Any party

presenting the proper bearer token can be authenticated. The methodology avoids the use of cryptographic keys, and requires protection from token disclosure during transit and storage. OAuth assumes a secure exchange of credential validation information between end points, specifically an OAuth client and server, prior to operation.

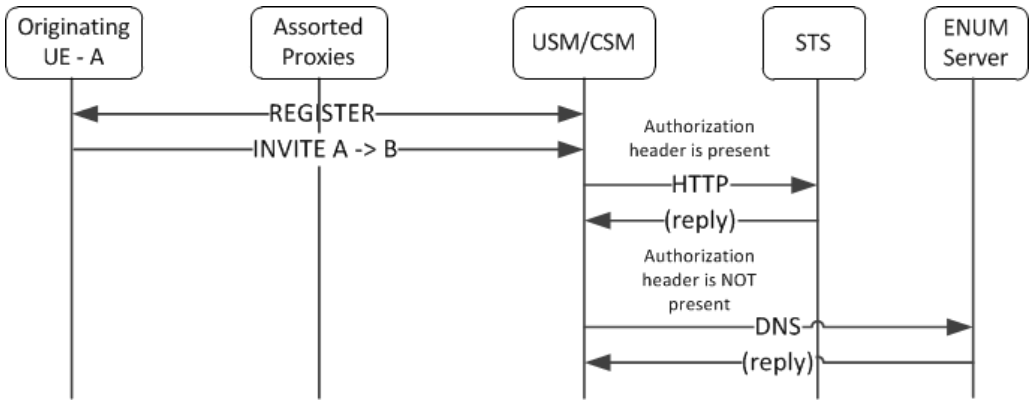
The Oracle USM implements OAuth in compliance with RFCs 6749 and 6750. OAuth typically requires deployment-specific compliance beyond RFC compliance. The Oracle USM allows for typical deployment environments via configuration.

You configure the Oracle USM to use OAuth by creating OAuth profiles and applying them globally and/or to specific interfaces. Interface profiles take precedence. In addition, you specify the server that the Oracle USM must contact for authentication using Oracle 's Session Processing Language (SPL). Note that this functionality requires that you upload a script file, provided by Oracle , and run an ACLI command to configure the script to use your server. This script file is referred to as a plug-in.

OAuth Operation

The Oracle USM assumes a client requesting OAuth authentication has previously acquired a token from its authorization server. This and token refresh procedures are performed outside of the scope of Oracle USM operation. Within the context of OAuth processes, the Oracle USM performs the functions of a resource server only.

Upon receiving applicable requests from a UA, the Oracle USM attempts to authenticate using OAuth or Digest. If the request includes a bearer string and a token, the Oracle USM initiates OAuth authentication by sending an HTTP query (GET request) to a Secure Token Service Server (STS), as shown below.



If the STS returns a 200 OK, the Oracle USM proceeds with authentication. If the STS does not reply or is not reachable, the Oracle USM replies to the UA with a 500 Internal server error. If the STS replies with any message other than a 200OK, the Oracle USM replies to the UA with a 403 Forbidden.

The Oracle USM uses the User ID within the STS 200OK to authenticate and authorize service, as follows:

- For register requests, the system compares this to the user ID field in the TO header of the SIP request.
- For other requests, the system compares it to the user ID field in the FROM header.

If the user ID matches, the Oracle USM proceeds with authentication. If this procedure concludes with matches, the Oracle USM provides service to the UA. If not, it replies to the UA with a 403 forbidden.

Note the following operational caveats:

- If the bearer token is present, but the Oracle USM is not configured for OAuth, the Oracle USM responds with a 500 internal service error message. Such misconfiguration includes the OAuth SPL plugin, described below, being absent or disabled.
- If the bearer token is not in the request, the Oracle USM proceeds with SIP digest authentication.
- If the request arrives over a secure channel, the Oracle USM follows the procedure defined by the applicable sip-authentication-profile, which may not require authentication.

Configuring OAuth Support

Configuring the Oracle USM for OAuth support consists of:

- Creating one or more OAuth profiles
- Applying each profile globally or to sip-interfaces

To configure an OAuth profile:

1. From superuser mode, use the following command sequence to access http-config element and define your profile.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# http-config
   ORACLE(http-config)# sel
   ```

2. Name your profile for reference within your configuration.
3. You can set the host parameter to an FQDN or an ip address. If you use an FQDN, the Oracle USM resolves it to an ip address when the configuration is loaded.
4. The num-connections parameter specifies the number of connections to be setup with the STS server. The Oracle USM establishes this number of connections when it boots or when the configuration is activated.
5. The max-outstanding-msgs parameter specifies the maximum number of outstanding messages per connection. An outstanding message is a request from the Oracle USM that has not had a response. When reaching this threshold, the Oracle USM stops sending requests on the connection until the number of outstanding messages falls back below this maximum. If the max-outstanding-msgs parameter is set to 1, the Oracle USM waits for a response before sending another request on a connection.
6. The Oracle USM uses the port parameter differently depending on whether the host parameter is configured as an IP address or an FQDN. If the host parameter is an FQDN name, the Oracle USM performs a lookup at a DNS server. In this case, you may or may not configure a port. If you configure the port parameter, the Oracle USM uses the configured port ignoring any port specified in the record returned by the DNS server. If you do not configure the port parameter, the Oracle USM uses the port returned by the DNS server. If the host parameter is set to an ip-address, you must configure the port parameter and the Oracle USM always uses the port parameter's value.

Enabling the SPL Plug-in

Enabling the SPL plug-in is a four step process.

1. Upload the SPL plug-in to a Oracle USM.

2. Add the SPL plug-in to the Oracle USM configuration.
3. Execute the SPL plug-in on the Oracle USM.
4. Synchronize the plug-in across HA pairs.

Uploading the Plug-in

The plug-in must be manually FTPed to the Oracle USM's /code/spl directory using any CLI or GUI-based FTP or SFTP application. The Oracle USM's FTP/SFTP server may be reached from the system's wancom or eth0 management physical interface.

Adding the Plug-in to Your Configuration

The plug-in must be configured in the spl-config configuration element. If multiple plugins are configured on the Oracle USM, the plug-ins are executed in the order of configuration.

To add the SPL Plugin to the configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **spl-config** and press Enter.

   ```
   ORACLE(system)# spl-config
   ORACLE(spl-config)#
   ```

4. Type **plugins** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMESYSTEM(spl-config)# plugins
   ACMESYSTEM(spl-plugins)#
   ```

5. Type **name**, a <space>, and the name of the SPL plug-in file**.**

   ```
   ACMESYSTEM(spl-plugins)#name SMXOauth2.spl
   ```

6. Type **done** to save your work.

Executing SPL Files

There are two ways to execute SPL files:

1. Perform a **save-config** and **activate-config** after exiting the configuration menu.
2. Execute the **reset spl** command—all configured SPL files are refreshed by with the **reset spl** command. You can also refresh a specific file by typing **reset spl <spl-file>.**

> **Note:**
>
> Oracle suggests that scripts are only refreshed during planned maintenance windows.

If an SPL file exists in the /code/spl directory, but is not configured in the **spl-files** parameter, it will be ignored when the ACLI User Interface is booting.

Synchronizing SPL Files Across HA Pairs

When running in an HA configuration, both the active and the standby systems must have the same version of the SPL plugins installed. To facilitate configuring the standby system, you can executing the **synchronize spl** ACLI command (without any arguments) to copy all files in the /code/spl directory from the active system to the same directory on the standby, overwriting any existing files with the same name.

By adding the specific filename as an argument to the **synchronize spl** command, the individual, specified scripts are copied between systems. For example:

```
ORACLE#synchronize spl SMXOauth2.spl
```

The **synchronize spl** command can only be executed from the active system in a HA pair. There is no means to synchronize SPL files automatically during a save and activate of the Oracle USM.

To sychronize all SPL Plug-ins to the configuration:

1. In Superuser mode, type **synchronize spl** and press Enter.

   ```
   ORACLE# synchonize spl
   ```

Configuring the Plug-in Option

For the SPL to work, you must configure it to recognize your http-server configuration. You do this by setting the **sts-server** option at the sip-registrar. This option is required for plug-in functionality.

To configure the sts-server option:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Access the target sip-registrar by typing **sip-registrar,** selecting your registrar and pressing Enter.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)# select
   ```

4. Type **sts-server=<http-config>, where "<http-config>"** is the name of your http configuration element, and press Enter.

   ```
   ACMESYSTEM(sip-registrar)# spl-options sts-server=http-server1
   ```

5. Type **done** to save your work.

Message Routing

The Oracle USM performs routing in two ways depending on the routing precedence parameter in the sip registrar. Routing precedence can be set to either **registrar** (ENUM) or **local policy**. Routing precedence is set to registrar by default.

Registrar routing uses the configured SIP registrar/ENUM server for destination address queries. Local policy routing lets you configure routing decisions within the Oracle USM's local policy routing functionality.

If the Oracle USM is performing any services for the call it performs those services prior to routing.

If, for any reason, the Oracle USM is unable to proceed with routing a request, it replies to the station that sent the request with a 4xx response.

Registrar Routing

When the routing precedence parameter is set to **registrar**, the Oracle USM is using the ENUM server as a resource within the context of its routing decisions.

When an INVITE arrives, the Oracle USM issues an query to the ENUM server. If the query's reply includes a match, the Oracle USM proceeds by forwarding the message via the information in the reply.

Note that you can configure the Oracle USM to fallback to a local policy lookup if the lookup via the registrar fails. Configure this by adding the **fallback-to-localpolicy** option to the sip-registrar configuration element.

For situations where the database routing decision needs to be done in lieu of the default, you can set routing precedence to local-policy. Note that you can configure a routing entry that points to an HSS by setting a policy attribute with a next-hop of enum:<server-name> within the local-policy.

Default Egress Realm

The sip registrar configuration element should be configured with a default egress realm id. This is the name of the realm config which defines the IMS control plane through which all Oracle USMs, ENUM servers, and other network elements communicate and exchange SIP messaging. It is advisable to configure this parameter in order to ensure well defined reachability among Oracle USMs.

SIP Registrar

To configure a SIP registrar configuration element for message routing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)#
   ```

4. Type **select** and choose the number of the pre-configured sip registrar you want to configure.

5. **routing-precedence**— Set this to either **registrar** or **local-policy** depending on your deployment.

6. **egress-realm-id**—Enter the default egress realm for Oracle USM messaging.
7. Type **done** when finished.

Segmentation of ENUM Zones

The Oracle USM supports operations with an ENUM root server, from which it can obtain partition delegation information and dynamically reach delegated authoritative ENUM servers for access information by ENUM zone. Obtaining this information is dynamic and supports query re-direction to other relevant authoritative servers the Oracle USM may learn from the root server. This configuration applies only to DDNS-based registration deployments. The user configures the **ddns-ns-caching** option to enable the Oracle USM for dynamically learned ENUM server caching.

For DDNS queries, the Oracle USM uses a manually-configured ENUM server configuration to reach a DNS root server. The root server can respond with the authoritative name server that manages the zone. The Oracle USM establishes dynamic entries in its ENUM server cache, with detail about the zones those servers service. This allows it to issue subsequent lookup queries directly to the correct server based on a match between telephone number and zone prefixes.

This feature assumes ENUM servers that are bind 8.0 compliant and return authority sections compliant with RFC 1304, section 4.3.2.

Additional operational details include:

- Resolution lookups are longest match, providing the Oracle USM with the ability to send a query to authoritative servers servicing zones and sub-zones.
- Dynamic server timeouts are equal to the TTL values received in the name server's resource record, provided in the NS response. The Oracle USM removes dynamic servers, as well as any associated ACL entries upon timeout.
- Redundancy for DDNS name server lookups is round-robin, triggered by standard DNS query timeout procedures, and following the order the system learns servers for the target zone.
- If a dynamic server fails to respond to a DDNS lookup, the Oracle USM removes that server from the lookup list. If there are no further servers in the list, the Oracle USM sends a 404 back to the station that originated the request. The Oracle USM then starts any subsequent lookups for that zone by querying the root server.
- Multiple NS records are supported. The Oracle USM creates NS records for the same zones in the order provided to it to establish name server redundancy.
- Should the Oracle USM need to obtain a resolution for a prefix that is unknown, the Oracle USM queries the original root server again.
- The Oracle USM removes any dynamically-learned name server from a query list when any query fails. That server is only added back to a list when it is dynamically re-learned.
- Should the root server become unavailable, the Oracle USM sends applicable queries to the next server in your manually-configured ENUM list.
- The Oracle USM does not execute its configurable health-query check processes with dynamically learned name servers.
- With respect to high availability deployments, the server list is not maintained on a standby Oracle USM. The ENUM configuration, however, is maintained on the standby via configuration synchronization. In the event of a failover, the standby Oracle USM learns all ENUM servers using the root server as a starting point.

- This **ddns-ns-caching** function interoperates with the Oracle USM's alphanumeric user name function. The zone match is done on the 3-bit checksum of the alpha-numberic user.

> **Note:**
>
> Name server records must present the Oracle USM with IP addresses for the server. Name server records using FQDNs are not supported.

Obtaining Information about Dynamic ENUM Servers

The Oracle USM provides ACLI commands that display real-time information about ENUM server interaction, including dynamically learned servers. The **show enum** command includes a section of statistics on cached entries

```
ORACLE# show enum stats localenum
 Parameter-> localenum lastArg  3 enumStatsType  0

SIP ENUM Statistics:
16:01:51-48
ENUM Agent localenum
                          -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Queries                -       -       3          3       3       -
  Successful           -       -       3          3       3       -
  NotFound             -       -       0          0       0       -
  TimedOut             -       -       0          0       0       -
  Bad Status           -       -       3          3       3       -
  Other Failures       -       -       0          0       0       -
Transactions           0       1       4          4       4       1
Cache Hits
  Successful           -       -       0          0       0       -
  NotFound             -       -       0          0       0       -
Cache Entries
  Successful           0       0       0          0       0       0
  NotFound             0       0       0          0       0       0
DropdCacheEntries      -       -       0          0       0       -
```

When appended with the **cache-console** argument, **show enum** displays detailed information about dynamic server configuration and status.

```
ORACLE# show enum status all cache-console
Showing 2 Enum Agents: state=Active cache
--------------------------------------
Enum Agent:       localenum
  Realm:          net192.4
  EnumServers:    192.168.53.199
  Query Timeout:  11
  Lookup Length:  3
  Health Query:   '' every 0 sec
  Failover To:
  IncludeSrcInfo: disabled
  Options:
  RecursiveQuery: disabled
DNS Agent:        EnumAgent[0x3b75a628(3)] ENUM w/o-stats
  Domain:         bogus.gy
  Local Address:  [256:0]192.168.53.170
  Service Types:  E2U+sip sip+E2U
```

```
  Trans ID:       7
  Cache Size:     0(0) inact-tmr=0 cache-addl=no
  Max Resp Size:  512
  Query Method:   hunt
servers:
    1=[256:0]192.168.53.199:53 OK
dynamic servers:
    1=[256:0]192.168.53.205:53 OK
```

Configuring Support for DDNS Server Caching

Use the procedure below to enable the Oracle USM to cache and manage DDNS servers for the purpose of efficiently forwarding ENUM queries to the servers responsible for the DNS zones identified in applicable SIP requests.

1. From superuser mode, use the following command sequence to access **enum-config** configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# enum-config
   ```

2. Use the **select** command to access the **enum-config** element or create a new one for the target ENUM root server.

3. Enable the **ddns-ns-caching** option.

   ```
   ORACLE(enum-config)# +option ddns-ns-caching
   ```

4. Use **done**, exit configuration mode, and run **verify-config** to complete DDNS cache support configuration.

   ```
   ORACLE(enum-config)# done
   ORACLE(enum-config)# exit
   ORACLE(session-router)# exit
   ORACLE(configure)# exit
   ORACLE# verify-config
   ----------------------------------------------------------------------
   Verification successful! No errors nor warnings in the configuration
   ORACLE#
   ```

Tel-URI Resolution

The Oracle USM can initiate number resolution procedures for requests that have tel-URI or SIP-URI (with user=phone) numbers in the R-URI. It does this by querying number resolutions services, including the local routing table(s) or ENUM server(s) to resolve the R-URI to a SIP URI. In addition, the original R-URI may not include a full E.164 number. As such, you can also configure the Oracle USM to perform a number normalization procedure and ensure it presents a full E.164 number for resolution. Upon successful resolution, the Oracle USM proceeds with ensuing signaling procedures.

To configure the Oracle USM to perform these lookups, you create applicable **local-routing-config** or **enum-config** elements and set an option within the **sip-registrar** that specifies a primary and, optionally, a secondary l**ocal-routing-config** or **enum-config** that the sip-registrar uses for LRT or ENUM lookups. If there is no ENUM configuration on the sip-registrar, the Oracle USM forwards applicable requests to a border gateway function via local policy.

Refer to the *Oracle Communications Session Border Controller ACLI Configuration Guide*, Session Routing and Load Balancing chapter for complete information on how to configure a **local-routing-config** and an **enum-config** elements.

Number Lookup Triggers

Use cases that are applicable to number lookups and the associated Oracle USM procedures include:

- Request from the access side:
  1. The Oracle USM performs originating services
  2. If the R-URI is a tel-URI or SIP-URI (with user=phone) and is not in the Oracle USM cache, it requests e.164 resolution from the ENUM server(s).
- Request from core side including request for originating services:
  1. The Oracle USM performs originating services
  2. If the R-URI is a tel-URI or SIP-URI (with user=phone) and is not in the Oracle USM cache, it requests e.164 resolution from the ENUM server(s).
- Request from core side, for terminating services only:
  1. If the R-URI is a tel-URI or SIP-URI (with user=phone) and is not in the Oracle USM cache, it performs a NAPTR lookup.
  2. If the reply indicates the tel-URI or SIP-URI (with user=phone) is not provisioned, the Oracle USM requests e.164 resolution from the ENUM server(s).

Actions Based on Lookup Results

The Oracle USM forwards to the resultant SIP-URI under the following conditions:

- The SIP-URI is in the Oracle USM cache, in which case the Oracle USM performs terminating services.
- The SIP-URI is not in the Oracle USM cache, and the Oracle USM is configured to service the returned domain.
  In this case, the Oracle USM performs the following:
  1. The Oracle USM issues an LIR for the SIP-URI.
  2. The Oracle USM forwards the message to the correct S-CSCF.
- The SIP-URI is not in the Oracle USM cache, and the Oracle USM is not configured to service the returned domain.
  In this case, the Oracle USM performs refers to local policy to forward the message via local policy.

PSTN Breakout Routing

The Oracle USM complies with RFC 4694 for operation with request-URIs that include carrier identification code/route number/number portability database dip indicator (cic/rn/npdi) information and routes those requests according to the rn information. The routing process includes utilization of local policy configured to break the request out of the home network via gateways such as a BGCF.

The Oracle USM does not validate any rn or cic information. Instead, it simply routes the request. Note that the Oracle USM uses cic information instead of rn if both are present in the request. RFC 4694 compliant circumstances under which the Oracle USM does not use rn, cic and npdi information include:

- Invalid routing information, including rn present, but npdi missing.

- Invalid routing information, including npdi present, but rn missing.
- Request uses a `sip-URI` presented without `user=phone`.

If the request includes originating services as well as cic/rn/npdi information, the Oracle USM performs those services rather than break out. If, after completing originating services, the request still includes cic/rn/npdi information, the system performs this breakout.

Primary and Secondary ENUM Configs

For the purpose of redundancy, the Oracle USM allows you to configure these number lookups to use a backup resource in case the lookup at the primary fails. Such scenarios include losing contact with the primary ENUM/LRT server config (query time-out) and the entry is not found at the primary (LRT or ENUM).

To apply primary and secondary number lookup resources to a sip-registrar:

1. From superuser mode, use the following command sequence to access the sip-registrar element and select the registrar you want to configure.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)# select
   ```

2. Specify the resources to use with the options command.

   Prepend the option with the + character if you have multiple options configured that you want to retain. Running the command without the + character causes the system to disable any previously configured options.

   To specify primary and secondary ENUM servers:

   ```
   ORACLE(sip-registrar)# options +e164-primary-config=enum:<enum-config name>
   ORACLE(sip-registrar)# options +e164-secondary-config=enum:<enum-config name>
   ORACLE(sip-registrar)# done
   ```

   To specify primary and secondary LRT resources:

   ```
   ORACLE(sip-registrar)# options +e164-primary-config=lrt:<lrt-config name>
   ORACLE(sip-registrar)# options +e164-secondary-config=lrt:<lrt-config name>
   ORACLE(sip-registrar)# done
   ```

   Bear in mind that an enum-config can reference multiple servers. When the Oracle USM references an enum-config, queries follow the normal enum-config sequence, checking each referenced server in order. If the lookup is not successful at the primary, the Oracle USM checks the servers in the registrar's e164-secondary-config.

   In addition, each enum-config may refer to a different top-level-domain. This allows you to configure the Oracle USM to successfully perform lookups within two domains.

Licensing and Database Registration Limits

The Oracle USM (and the Oracle Communications Session Border Controller) limit the number of unexpired registration cache entries globally. The total number of system registrations is configured with the registration cache limit parameter in the **sip config** configuration element.

The Oracle USM also limits the number of registration cache entries that were obtained from a User Subscriber Database; only REGISTERs that prompted the database query are counted here. As User Subscriber Database entries are added and removed, this counter is updated

accordingly. Note that it is the actual number of SD-contacts that count against the license limit. Discrete database registration license values range from 20,000 through 500,000 in increments of 20,000.

When a registering contact is rejected because it will exceed one of these limits, the Oracle USM sends a 503 message to the registering endpoint.

Refer to the *Oracle Communications ACLI Configuration Guide*, chapter 2 Getting Started, Software Licensing section for how to install a license.

Database Registration Limit Alarm

By default, a major alarm is enabled when 98% or more of the licensed number of Database Registrations are used. This alarm is cleared when the number of database registrations falls below 90%. You can configure minor and critical alarms when crossing configured thresholds and you can also reassign the major alarm. This is configured in by creating a system-config > alarm-threshold sub element with type of **database-registration**.

Extended ENUM Record Length

In some cases, when a user registers a contact from a UA, the Contact: header's contents are too long to be inserted into the DNS-based user database as presented in the NAPTR record.

To mitigate this, the Oracle USM stores contact-related metadata, i.e. parameter key and value pairs, in a related TXT record.

If the contents of the Contact: header do not exceed 255 bytes, there is generally no need to extract the metadata to store in an additional TXT record, so the Oracle USM will not enact this process.

NAPTR and TXT Record Creation and Association

When a user initially registers to the Oracle USM, the DDNS update sent to the user database will include a NAPTR record and a TXT record. The NAPTR record contains the Contact: header's SIP URI as a regexp replacement. All URI and header parameters from the Contact: header are excluded from the NAPTR record and are inserted into the accompanying TXT record.

The NAPTR record and the TXT record both contain a Oracle USM-generated key that binds the Contact metadata and the Contact URI sent in the separate records. This common key indicates the data in the NAPTR and TXT record belong to the same registering contact, and is used to recreate the Contact: header for later use. The format of the common key is:

```
p-acme-ckey=<SD-Core-IP>:<integer id>
```

The <SD-Core-IP> is the IP address from which the Oracle USM communicates with the DNS server. The <integer id> enumerates each contact. Contacts are enumerated in the integer-id element because they can be non-unique when a user behind a NAT registers more than one contact from behind the NAT. For example:

```
    TXT "p-acme-ckey=172.16.101.61:1" "$key=value" "^key=value"
NAPTR 1 1 "u" “E2U+sip""!^.*$!sip:642-10u72@172.16.101.61:5060\;p-acme-
key=172.16.101.61:1!"
```

In the previous example, the key=value pair represent parameters in the Contact header and Contact SIP URI. A $key= indicates the parameter existed in the Contact: SIP URI. A ^key=

indicates the parameter existed in a Contact: header parameter. The Oracle USM uses this convention to reconstruct the parameters' placement in the original Contact: header.

NAPTR Record Format

Oracle USMs learn that an associated TXT record exists for the from a NAPTR lookup for the queried SIP URI.

In the DDNS update, the Oracle USM indicates an associated TXT record by setting the NAPTR resource record with:

- m in the flags field (in addition to the u flag)
- E2U+sip:contact in the service type field

When no additional metadata and no parameters need to stored in ENUM server, the flags field contains u.

TXT Record Retrieval

A Oracle USM performs a separate query to the ENUM server for the TXT metadata records when it detects the 'm' flag in a NAPTR result (and the one-query-txt-naptr option is not configured). You can set the one-query-txt-naptr option to enabled in the enum-config configuration element to force the Oracle USM to request the TXT and NAPTR records both in one query from the ENUM server.

Once the Oracle USM receives the metadata stored in the TXT records, it restores the header and URI parameters that were present on the original registration.

Requirements

BIND 9.8.1-P1 or later hosting the ENUM server supports this feature.

SIP User Parts - RFC 3261 Character Set Support

RFC 3261 specifies the range of characters allowed in a user-part, all of which are supported by the Oracle USM. ENUM databases, however, do not support all these characters.

By default, the Oracle USM presents SIP messages to your DNS server unchanged, assuming that your deployment uses telephone numbers only as URI user parts. You configure the Oracle USM to support the entire range of characters allowed in user parts by enabling it on your ENUM server configuration. Upon configuration, the Oracle USM grooms the user part appropriately for the ENUM server.

Encoding Alpha-Numerics

If your deployment uses non-numeric characters in user parts, you can explicitly enable the Oracle USM to change the user part to be compatible with the ENUM database. In these cases, the Oracle USM encodes SIP message user parts using a proprietary encoding. The Oracle USM provides the encoded string to the DNS server, which then creates the applicable record(s). The Oracle USM decodes these strings as necessary during subsequent interactions with the DNS server and the UAs.

In some cases, deployments include SIP messaging in your environment that does not require encoding. These cases include messages with SIP URI user parts composed of:

- tel-uri
- sip-uri with all numeric characters
- sip-uri with all numeric characters except for a leading +

In these cases, the Oracle USM does not encode these user parts. In addition, for all-numeric user parts proceeded by the + character, the Oracle USM strips the + character, reverses the digits, separates the digits with periods, and sends the message to the ENUM server with the user part unencoded.

Multiple DNS Zone Support

For smaller deployments, one can assume a single DNS zone within which all applicable UEs reside. Large deployments, however, may use multiple DNS zones, allowing the network administrator to segregate and more easily manage large numbers of UEs.

When the Oracle USM encodes the user part, it performs a modulo 1000 operation on a 24-bit checksum of the user part and appends the resulting 3 digits to the encoded user part. You can configure your DNS zones for these digits to organize users into zones.

For example, encoding the user part acme_user gives the string acmeX5Fuser.2.7.7. Assuming a domain of acme-ims.com, the ENUM entry for this user would be acmeX5Fuser.2.7.7.acme-ims.com. Applicable zone configuration can use the .2.7.7 portion of this string to determine this user's zone.

Alpha-Numeric Name Support

To enable to use of alpha characters in SIP message user parts for a given sip-registrar:

1. From superuser mode, use the following command sequence to access sip-registrar element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# enum-config
   ORACLE(enum-config)# select
   ```

   Select the enum-config you want to configure.

2. Specify alpha-numeric name support for this enum-config and specify the number of DNS zones in your deployment.

   ```
   ORACLE(enum-config)# alpha-numeric-user-support enabled
   ORACLE(enum-config)# done
   ```

   Note that the parameter alpha-numeric-user-support is RTC supported.

Configuring SIP Ping OPTIONS Support

You can configure the Oracle USM to respond to SIP ping OPTIONS. This support is typically configured on an S-CSCF so it can respond to pings OPTIONS sent by a P-CSCF:

To configure an SIP Options Ping response support:

1. From superuser mode, use the following command sequence to access ping-response command on a sip-interface element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# sel
```

2. Enable the support with the ping-response command.

```
ORACLE(http-config)# ping-response enabled
ORACLE(http-config)# done
```

ping-response—Enable ping-response to allow your device to respond to ping OPTIONS. For example, this feature is useful within hybrid deployment environments on a P-CSCF as a means of verifying the S-CSCF's availability. This configuration allows the S-CSCF to respond to SIP ping OPTIONS.

8

Local Subscriber Tables

Local Subscriber Table

A local subscriber table (LST) is an XML formatted file that contains one or more usernames associated with a hash as encrypted or plaintext. The LST is saved locally on the Oracle USM's file system.

LSTs enable a standalone Oracle USM node or high-availability (HA) pair to forego relying on an external user database. Thus the Oracle USM does not need to communicate with a server to authenticate users. This can eliminate the operational complexity of deploying a highly available credential storage system.

LST Runtime Execution

The LST is loaded on boot up when the configuration is appropriately set. Incoming messages thereafter can then be authenticated based on the credentials in the LST. If the Oracle USM can not load an LST file, three things occur:

1. The following log message is recorded at the NOTICE level:

   ```
   LST [table-name] was not loaded - [filename] has error loading XML file
   ```

2. The message stated above is printed on the ACLI.
3. A 503 Response is returned to the UA that sent the initial REGISTER message to the Oracle USM.

LST Configuration

To configure the Oracle USM to use LSTs for authentication, you need to create a local subscriber table configuration element that identifies that LST. You then need to set the sip authentication profile configuration to reference that LST configuration so that when messages requiring authentication are received and processed by a sip registrar configuration element, the Oracle USM will use the identified LST for authentication.

In a local subscriber table configuration, you must define an object **name**, identify the specific LST **filename** (and path). If the filename is entered without a path, the Oracle USM looks in the default LST directory, which is /code/lst. If the LST file is located elsewhere on the Oracle USM, you must specify the filename and absolute path. For example /code/path/01302012lst.xml.

The corresponding sip authentication profile must be set to use the **local subscriber table** configuration element you just created. First set **credential retrieval method** to **local**, set the **digest realm** appropriately (this is required for authentication), and finally set the **credential retrieval config** parameter to the **name** of the local subscriber table configuration element that you just created. At this point you may save and activate your configuration.

Unencrypted passwords for each user in the table is computed with the MD5 hash function as follows:

```
MD5(username:digest-realm:password)
```

ACLI Instructions

LST Table

To configure the Oracle USM to use an LST:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-subscriber-table** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# local-subscriber-table
   ORACLE(local-subscriber-table)#
   ```

   You may now begin configuring the local subscriber table configuration element.

4. **name**—Enter the name of this local subscriber table configuration element that will be referenced from a SIP registrar configuration element.
5. **filename**—Enter the filename that describes this LST XML file. If no path is given, the Oracle USM looks in the /code/lst directory. You may provide a complete path if the file is located elsewhere.
6. **secret**—Enter the PSK used in encryption and decryption of the passwords in the XML file. Once saved, this value is not echoed back to the screen in plaintext format. See LST Subscriber Hash and Encryption.
7. Type **done** when finished.

SIP authentication profile

To configure the Oracle USM to utilize an LST, continuing from the previous step:

1. Type **exit** to return to the session router path.
2. Type **sip-authentication-profile** and press Enter.
3. Type **select** to choose the existing sip-authentication-profile configuration element you wish to use LST for authentication.

   ```
   ACMESYSTEM(sip-authentication-profile)# select
   <name>:
   1: name=sipAuthSMX1 digest-realm=acme.com credential-retrieval-method=loacl
   selection: 1
   ACMESYSTEM(sip-authentication-profile)#
   ```

4. **digest-realm**—Enter the digest realm used for authenticating here.
5. **credential-retrieval-method**—Set this parameter to **local** to use an LST.
6. **credential-retrieval-config**—Enter the name of the LST configuration you just configured.
7. Type **done** when finished.

LST Redundancy for HA Systems

LSTs must be synchronized between redundant nodes to ensure that the standby node contains identical LST files. You can either SFTP the same LST file to both the active and standby node, or you can use the synchronize command. The **synchronize** command is always executed from the active system. It copies the specified file from the active to the standby node placing the copy in the same file location on the standby node. Use the **synchronize lst** command as follows:

```
ACMESYSTEM# synchronize lst file.xml
```

> **Note:**
>
> The synchronize command does not reload the LST files.

Reloading the LST

After copying a new LST file to the Oracle USM (and its standby peer), you can reload this newer file from the ACLI using the **refresh lst** command. For example:

```
ORACLE# refresh lst <local-subscriber-table name>
```

Using the **refresh lst** command selects the LST by name to refresh. Alternatively, saving and activating the configuration will reload the configuration as well and should be used when configuration parameters have also changed.

> **Note:**
>
> In an HA pair of Oracle USMs, you must independently execute the refresh lst command on both the active and standby systems.

LST File Compression

To save local disk flash space, you can compress the LST XML file using .gz compression. The resultant file must then have an .xml.gz extension.

LST File Format

The LST file format is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<localSubscriberTable encrypt-algo="aes-128-cbc">
<subscriber username="alice@apkt.com" hash="02:5E:78:D8:7E:75:A3:39"
encrypted="true"/>
<subscriber username="bob@apkt.com" hash="bc4b2a76b9719d911017c59"
encrypted="false"/>
<subscriber username="acme@apkt.com" hash="5d41402abc4b2a76b9719d9"
encrypted="false"/>
</localSubscriberTable>
```

The LST file's elements are as follows:

localSubscriberTable

This is the head element in the XML file. Each file can have only one head element. The following attribute is found in this element:

- encrypt-algo—This indicates the algorithm type used to encrypt the hash in the XML file. The key for this encryption will be a preshared key and is configurable in the local subscriber table configuration element with the secret parameter.
- The value in this element is for display purposes only.
- Currently AES-128-CBC is the only supported encryption algorithm.

subscriber

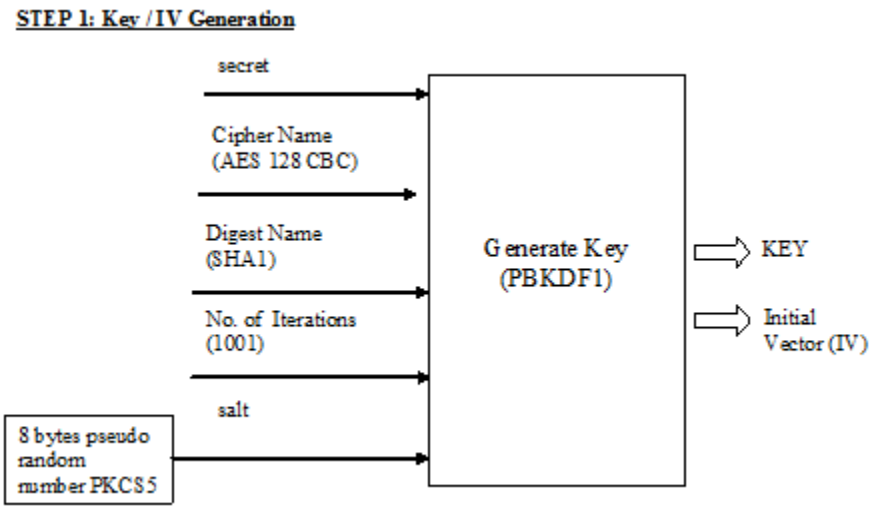
This element has the subscriber information. And has the following 3 attributes:

- username—The value given in the username attribute must be same as the username that will be sent in the Authorization Header in the Request message from the users. Refer RFC 2617 Http Authentication for details.
- hash—The hash provided in the XML must be an MD5 hash of the Username, digest-realm and the password of the user. This is same as the H(A1) described in RFC 2617. hash = md5(username:digest-realm:password)
- encrypted—The encrypted flag indicates if the "hash" given in the XML file is encrypted or not

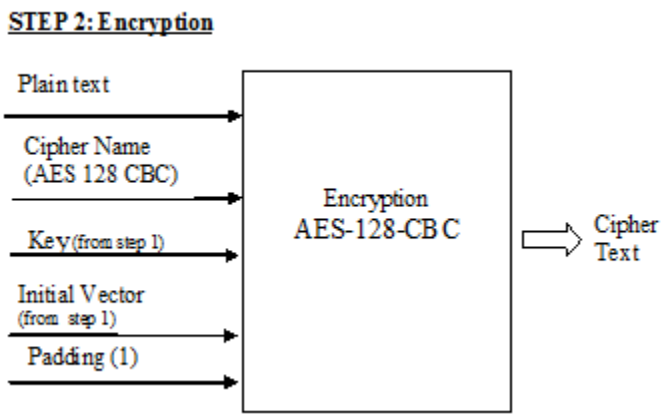
LST Subscriber Hash and Encryption

You may additionally use AES-128 CBC to encrypt the hash in the subscriber element in the LST XML file. The PSK used for encryption is configured in the **secret** parameter and an 8-byte pseudo random number is used as the salt. The LST file must set the encrypted attribute per subscriber element to true. To derive the final encrypted data you place in the XML file, three steps are performed according to the following blocks. The output of the last step, Formatting final Encrypted Data, is inserted into the LST files, subscriber element’s hash value, when the encrypted attribute is set to true.

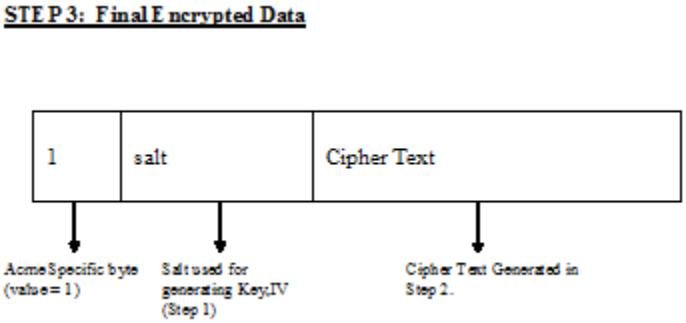
Key Initialization Vector



Encryption



Formatting final Encrypted Data

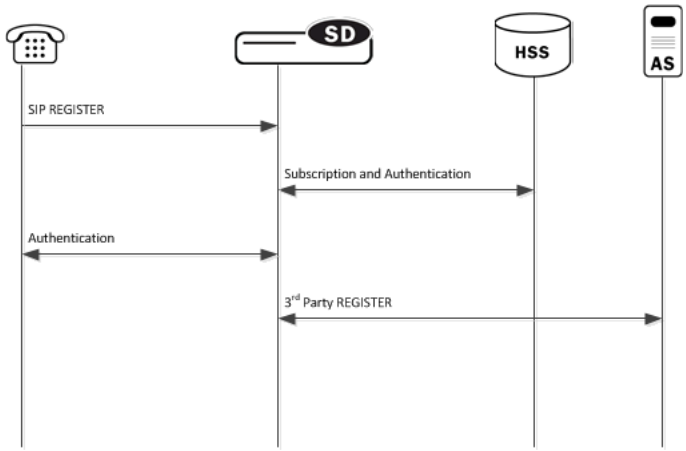


9

Third Party Registration

Third party registration support on the Oracle USM provides a mechanism for sending registration information to a third party server. An IM (Instant Messaging) server might be the recipient of a third party REGISTER message.

The Oracle USM accepts incoming REGISTER requests from UAs. After the UA has been registered with the Oracle USM, the Oracle USM sends a third party REGISTER message to a third party server.



The Oracle USM supports third party registration via two methods:

- For scenarios in which UAs receive iFCs from the HSS and the Oracle USM's default iFC configuration, the Oracle USM generates third party registration requests and responses for matching triggers in its iFC evaluation.
  Some third party servers may want the UA's entire original request to the Oracle USM and response from the Oracle USM to the UA provided to them. The Oracle USM supports these scenarios, in some cases requiring additional configuration.
- For scenarios in which the UA needs a third party registration that is not explicitly prescribed within iFCs, you can configure a third party server on the Oracle USM and achieve third party registration support.
  For these configurations, the Oracle USM attempts third party registration to those servers for all UAs that register via the applicable Oracle USM registrar.

For both methodologies, you must configure all third party servers as session agents.

Third Party Registrations via iFCs

The Oracle USM performs third party registrations based on the iFC downloaded for the user. If the filter criteria successfully evaluates to a third party server, a third party registration entry is dynamically added in the Oracle USM. The dynamic entry is automatically deleted if there are no more registrations being handled for that third party registration host.

When third party registration is performed by iFCs, the Oracle USM generates the registration messages as follows:

- The Contact: header is populated with the URI from the home server route configuration of the sip-registrar associated with the registration. If the home server route is left blank, the Oracle USM uses the IP address of the egress interface.
- The From: header of the new REGISTER message is the same as the FROM in the original message.
- The To: header of the new REGISTER message is the the same as the TO in original message (AOR).

Embedded REGISTER

As an option within standard iFC third party registration support, the Oracle USM supports 3GPP's methodology of embedding the original UE registration (and/or its response from the S-CSCF/Registrar) as a MIME body in the third party REGISTER sent from the S-CSCF to the third party server. This methodology, presented in 3GPP TS 23.218 and 29.228, uses an optional iFC extension ("IncludeRegisterRequest" and "IncludeRegisterResponse") that tells the third party server to expect the entire original REGISTER request and/or REGISTER 200OK in the mime of the third party REGISTER.

Implementation details for this methodology include the following:

- There may be further configuration required on the Oracle USM.
- The Oracle USM does not embed original registration requests or responses to any third party server outside its trust domain.
- The HSS or configured iFCs must be preconfigured for embedded third party registrations.

An HSS confiuration may not support the optional "IncludeRegisterRequest" and "IncludeRegisterResponse". For these cases, there is a Oracle USM configuration option that allows you to control this inclusion, as follows:

- If the iFCs specify inclusion in an environment where you do not want it, you can set a registrar option to never include the original REGISTER
- If the iFCs do not specify inclusion in an environment where you want it, you can set a registrar option to always include the original REGISTER.

You can set these options for either the third party register, the 200 OK, or both.

ACLI Instructions - Third Party Registration via iFCs

Session Agent

To create a session agent to represent the third party server:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **hostname**—Enter the name for this session agent.
5. **ip-address**—Enter the IP address for this session agent. This value must be the same as the registrar-host parameter in the third party regs configuration element to which this session agent definition corresponds.

   Continue configuring this session agent's parameters. Not all session agent functionality is applicable to the Oracle USM.
6. Type **done** when finished.

SIP Registrar

Option to set the SIP Registrar to perform embedded REGISTRATION support for third party registration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```
2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```
3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ORACLE(sip-registrar)#
   ```
4. Type **select** and choose the number of the pre-configured SIP registrar configuration element you want to configure.

   ```
   ORACLE(sip-registrar)# select
   name:
   1: registrar1
   selection:1
   ACMEPACKET(sip-registrar)#
   ```
5. **option +include-register-request**—Set this option to control SIP REGISTER embedding in the third party registration.

   ```
   ORACLE(sip-registrar)#options +include-register-request=true
   ```

   Set this option to true to always embed the original REGISTER in the third party registration.

   In some cases, the include may already be specified by the iFCs, even though you do not want it used. In these cases, configure the option to false

   ```
   ORACLE(sip-registrar)#options +include-register-request=false
   ```
6. **option +include-register-response**—Set this option to control SIP REGISTER 200 OK embedding in the third party registration the S-CSCF sends to the AS.

   ```
   ORACLE(sip-registrar)#options +include-register-response=true
   ```

   Set this option to true to always embed the original REGISTER in the third party registration 200 OK.

   In some cases, the include may already be specified by the iFCs, even though you do not want it used. In these cases, configure the option to false.

```
ACMEPACKET(sip-registrar)#options +include-register-response=false
```

7. Type **done** when finished.

Third Party Registration via ACLI Configuration

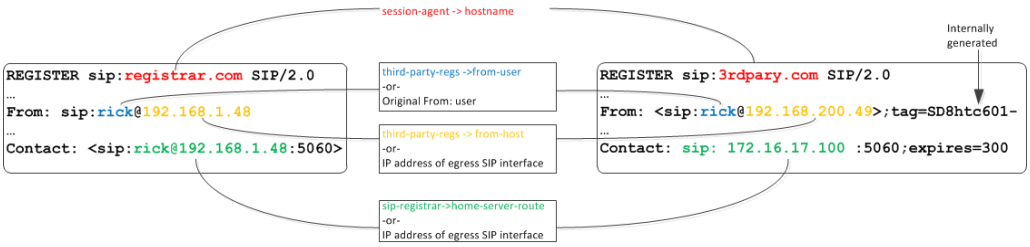
This section specifies the differences between Oracle USM third party registration support via iFC as oppsed to via ACLI configuration.

As is true of the method described above, third party registration is generated by the Oracle USM on behalf of the user in the To: header of REGISTER request.

When third party registration is generated by ACLI configuration on the Oracle USM, the registration messages are generated as follows:

- The request URI of the new REGISTER message uses the value of the hostname parameter in the session agent configuration element.
- The From: header of the new REGISTER message uses the value of the from-user parameter in the third party regs configuration element as the user portion of the URI. If the from-user parameter is left blank, the Oracle USM uses the user in the original From: header.
- The From: header of the new REGISTER message uses the value of the from-host parameter in the third party regs configuration element as the host portion of the URI. If the from-host parameter is left blank, the Oracle USM uses the IP address of the egress SIP interface as the host portion of the from header.
- The Contact: header of the new REGISTER message uses the home server route parameter in the sip registrar configuration element. If the home server route parameter is left blank, the Oracle USM uses the IP address of the egress interface.

See the following diagram:



Third Party Registration Server States

If the third party server does not respond to a REGISTER request, the Oracle USM adheres to standard SIP session agent retransmission/ timeout procedures. If the third party server is set to out of service, the Oracle USM attempts connectivity retry procedures. The retry procedures dictate that the Oracle USM periodically send a REGISTER message to the third party server to check if connectivity has come back. The time interval for checking connectivity to a third party server is set with the retry interval parameter. Retries continue forever or until the third party server responds. The retry mechanism may be disabled by setting the retry interval parameter to 0.

> **Note:**
>
> When using the ACLI generated third party registration method, the time interval for checking connectivity to a third party server is set with the retry interval parameter in the third party regs configuration element.
> When a third party server is out of service, the Oracle USM maintains a queue of outstanding third party registration requests. When the third party server returns to service, the Oracle USM gracefully flushes the queue of outstanding requests. This prevents a registration flood from being directed at the third party server .

Third Party Registration Expiration

The REGISTER message sent from the Oracle USM to the third party server uses the Expires: value returned from the User Subscriber Database or HSS. The third party server sends a 200 OK message containing Contact bindings and an expires value chosen by the third party server itself. The Oracle USM checks each contact address to determine if it created it. For those addresses it created (as SD-Contacts), the Expires value from the 200 OK is used as the final value.

Once the expires timer has reached half the expires period as returned from the third party server, the Oracle USM refreshes the registration.

If the third party server responds to a REGISTER Request with a 423 (Interval Too Brief) response, the Oracle USM updates the contact's expiration interval to the Min-Expires value of the 423 response. It then submits a new REGISTER Request with the updated expires value.

Defining Third Party Servers

To send third party registrations that are generated via ACLI configuration to a third party server, three configuration elements are required. The primary configuration element is the third party regs. One or more may be configured in order to send the REGISTER message to multiple registration servers. You need to configure a name and set the state to enabled. The registrar host must be configured to indicate the value to insert into the Oracle USM-generated request URI in the REGISTER message.

> **Note:**
>
> It is recommended that the list of third party registration servers be restricted to a maximum of 3.

A session agent needs to represent the third party server. Create a session agent as the third party server and note its name. Next, configure the registrar-host parameter with a session agent hostname in the third-party-reg configuration element. This specifies the session agent to be used as the registrar.

Finally, the address of the third party server must be added to the third-party-registrars parameter in the sip-registrar configuration element. This does not supercede any core Oracle USM Registrar functionality. It informs the Oracle USM of the third party server to send messages to after initial registration. Thus the value configured here must exist in the third-party-regs configuration element's registrar-host parameter list.

ACLI Instructions - Third Party Server Configuration

Recall that the configuration below is only required for scenarios in which the iFC does not explicity specify registration for the servers you configure below.

Third Party Registrar

To configure a third party server:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **third-party-regs** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# third-party-regs
   ACMEPACKET(third-party-regs)#
   ```

4. **state**—Set this to enabled to use this configuration.
5. **registrar-host**—Set this value to the complementary session agents' hostname parameter to include those session agents as third party servers. This parameter may be modified like an options parameter. This value also appears in the request URI of the outgoing REGISTER message being sent to the third party server.
6. **from-user**—Configure this parameter to be the user portion of the From: header of the outgoing REGISTER message being sent to the third party server. Leaving this blank sets the user portion that in the original From: header
7. **from-host**—Configure this parameter to be the host portion of the From: header of the outgoing REGISTER message being sent to the third party server. Leaving this blank sets the host portion to the Oracle USM's egress SIP interface.
8. **retry-interval**—Enter the number of seconds the Oracle USM waits before retrying a third party server after a failed registration. Enter **0** to disable this feature.
9. Type **done** when finished.

SIP Registrar

To indicate to a local SIP Registrar when and what third party server to send third party registrations to:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-registrar** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-registrar
   ACMEPACKET(sip-registrar)#
   ```

4. Type **select** and choose the number of the pre-configured SIP registrar configuration element you want to configure.

```
ORACLE(sip-registrar)# select
name:
1: registrar1
selection:1
ACMEPACKET(sip-registrar)#
```

5. **home-server-route**—Enter the value to insert into the REGISTER message's request URI as sent to the third party server. Leaving this blank uses the AoR (or To: header) in the original REGISTER message.
6. **third-party-registrars**—Enter the name of a third party regs configuration element registrar-host parameter to send third part registrations associated with that SIP registrar.
7. Type **done** when finished.

10

Oracle USM Supporting the IMS Edge

The ETSI TISPAN NGN defines several subsystems that make up the NGN architecture. The Oracle USM is an integrated session control, policy enforcement and media management solution that incorporates functional components of the IP multimedia subsystem (IMS), the Resource and Admission Control Subsystem (RACS) and functions necessary for connecting with other IP networks/domains.

The functions of the Oracle Communications Session Border Controller within the NGN architecture are divided into access/interconnect and core functions. This chapter addresses the access and interconnect functions. Oracle USM core functions are addressed in the Oracle USM Supporting the IMS Core chapter.

Access Border Functions

The Oracle USM is deployed as the access point between the core IMS network and UEs to deliver the functions defined in the TISPAN architecture as the P-CSCF, and A-BGF. These two functions can not be separated. The Oracle USM performs the following functions as the Access Oracle USM:

- P-CSCF functions
- Access Border Gateway Functions (A-BGF)
- Resource and Admission Control Functions (RACF)

P-CSCF Functions

The Oracle USM performs the following functions in the role of P-CSCF:

- Forwards SIP REGISTER messages and maintains a cached mapping of the user info and the UE's Address of Record (AoR), including the far-end NAT address in the case of hosted NAT traversal (HNT).
- Performs local emergency session handling—Local routing policy is used by the Oracle USM to identify emergency sessions and provide unique routing (e.g. can route to a dedicated S-CSCF function for emergency session handling).
- Operates as a UA (B2BUA) for generating independent SIP transactions for security purposes and handling of abnormal conditions.
- Offers current session timers which are used to monitor for media faults and abandoned calls.
- Generation of CDRs—The Oracle USM generates real-time accounting records via RADIUS.
- Authorization of bearer resources and QoS management—With integrated BGF capabilities, the Oracle USM allocates bearer resources (NAPT flows) and applies QoS policies (including packet marking) based on local policies and/or policies acquired via interaction with the A-RACF (PDF).

- Interaction with the A-RACF (PDF) for session-based policy enforcement and admission control—The Oracle USM PDF interface options include COPS and SOAP/XML.
- Traffic Policing—Traffic is policed at the session and media/transport layer. At the signaling layer, the Oracle USM polices at a number of levels including:
  - Capacity—Total number of concurrent calls to/from each realm
  - Session set-up rate—Maximum rate of call attempts to/from each signaling element
  - Signaling message rate—Each endpoint's signaling message rate is monitored and policed
  - Signaling bandwidth—each endpoint's signaling bandwidth is policed individually

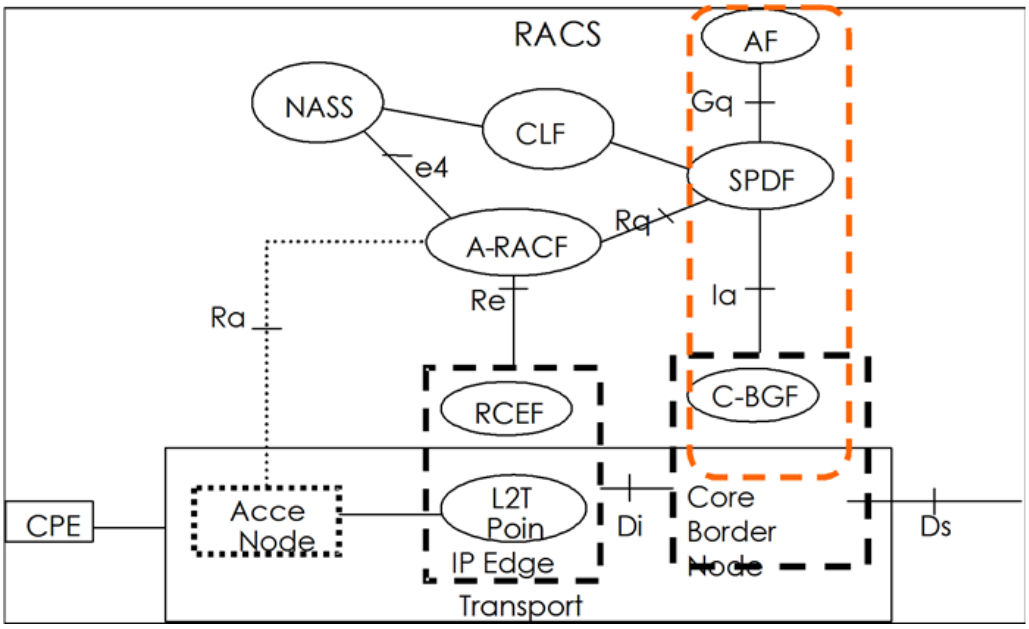
A-BGF Functions

The Oracle USM performs the following access-related BGF functions:

- Opening and closing gates/packet filtering—The Oracle USM opens and closes gates (media pinholes) on a session-by-session basis. Packet filtering rules include full source and destination IP address and port number.
- Per-session DiffServ or ToS marking—Media flows destined for the IMS core network can be explicitly marked using ToS or DiffServ. Media packets can be marked by VPN, by codec (voice, video) or by E.164 phone number prefix.
- NAPT-PT and topology hiding—The Oracle USM provides NAPT for all media flows associated with a session on a per session-basis. Double NATing, NATing both source and destination sides, is utilized to fully hide topology in each direction for RTP and RTCP. Local IP addresses and port resources are dynamically allocated from steering pools provisioned on the Oracle USM.
- Hosted NAT traversal—The Oracle USM supports HNT function that allows media flow traversal through the CPE firewall/NAT without upgrading the CPE equipment. The Oracle USM interacts with the endpoints to dynamically establish and maintain bindings in the CPE firewall/NAT that allow the signaled communications to pass through. The Oracle USM's registration management and media relay functions make CPE-based NATs transparent to the service delivery elements.
- Traffic Policing—Traffic is policed at the session and media/transport layer. At the signaling layer, the Oracle USM polices at a number of levels.
- Policing of Media (e.g. RTP & RTCP) traffic on a per-flow basis—CBR policing is applied to each flow based on offered and negotiated media codecs.

Resource and Admission Control (RACS) Functions

The figure below illustrates the mapping of Oracle USM functions to the RACS functional model. In this model, the Oracle USM incorporates the Application Function (in the case of IMS this is the P-CSCF function), the SPDF (Service Policy Decision Function) and the Core Border Gateway function.



The Oracle USM, acting as the SPDF, interfaces with the PDF (A-RACF policy decision function) for resource authorization and admission control on a call-by-call basis. COPS is the supported PDF interface.

IMS Interconnect Border Functions

The Oracle USM is deployed at IP interconnect points between service providers to deliver the functions defined in the TISPAN architecture as the Interconnect Breakout Gateway Control Function (I-BGCF). The Oracle USM performs the following functions as the interconnect border Oracle USM:

- Interaction with I-BGF (including NAPT and firewall functions)
- Topology hiding-screening of signalling information

Oracle USM Access Interface Configuration

The Oracle USM supports two interface types. Oracle USM interfaces are core-side interfaces by default, and support IMS and OTT functions for the core. Access-side interfaces perform entirely different functions on the access-side of the Oracle USM. The user must identify access-side SIP interfaces and enable them for access and interconnect-related functions as presented below.

Further configuration may or may not be required for a specific function.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. Type **select** and the number of the pre-configured SIP interface you want to configure.

   ```
   ORACLE(sip-interface)# select 1
   ```

5. **ims-access**—Enable IMS access-side functionality on this SIP interface. The default value is **disabled**. Valid values are:

   - enabled | disabled

     ```
     ORACLE(sip-interface)# ims-access enabled
     ```

6. Save your work using the ACLI **done** command.

Wildcard PUI Introduction

The Oracle USM supports wildcard Public User Identities (PUI). This capability is most often used to streamline processing of REGISTER and INVITE messages between a PBX and an IMS core.

The HSS (Home Subscriber Server, an IMS network-wide database) is pre-provisioned with the extension numbers associated with the PBX's base telephone number. When the PBX registers its own base telephone number with the Oracle USM, that server downloads a wildcarded PUI -- a regular expression that describes all extension numbers associated with the registering PBX.

The Oracle USM constructs a registration cache entry to implicitly treat subsequent INVITEs from PBXs extensions that match the wildcard as registered endpoints.

A wildcarded PUI consists of a delimited regular expression located either in the user info portion of the SIP URI or in the telephone-subscriber portion of the Tel URI. The regular expression takes the form of an Extended Regular Expressions (ERE) as defined in chapter 9 of The Single UNIX Specification (IEEE 1003.1-2004 Part 1). The exclamation mark (!) serves as the delimiter.

For example, the following PUIs will match to the wildcard ERE "sip:chatlist!.*!@example.com".

```
sip:chatlist1@example.com
sip:chatlist2@example.com
sip:chatlist42@example.com
sip:chatlistAbC@example.com
sip:chatlist!1@example.com
```

Wildcard PUI Message Flows

A basic registration exchange with support for PUIs enabled.

From UA to Oracle USM

```
REGISTER sip:192.168.1.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-13794-1-0
From: <sip:7818888@hxu.com;tag=1
To: sut <sip:7818888@hxu.com>
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
Contact: <sip:7818888@192.168.1.231:5060>;expires=3600
Content-Length: 0
```

From Oracle USM to UA

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-13794-1-0
From: <sip:7818888@hxu.com;tag=1
To: sut <sip:7818888@hxu.com>;tag=1
Call-ID: 1-13794@192.168.1.231
CSeq: 1 REGISTER
Contact: <sip:7818888@192.168.1.231:5060>;expires=3600
P-Associated-URI: <sip:781!.*!@hxu.com>
Content-Length: 0
```

Incoming Request INVITE from Core

For the incoming request from the core to the Oracle USM, if reg-cache-route is enabled on ingress SIP Interface, the Oracle USM checks the registered P-CSCF-contact-URI in the top Route P-CSCF-contact-URI, instead of with request-URI.

```
INVITE sip:7816666@192.168.200.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.231:5060;branch=z9hG4bK-13800-1-0
From: sipp <sip:3054252165@192.168.200.231:5060;user=phone>;tag=1
To: sut <sip:7816666@hxu.com>
Call-ID: 1-13800@192.168.200.231
Supported: 100rel,timer,resource-priority,replaces
CSeq: 1 INVITE
Contact: sip:7816666@192.168.200.231:5060
Max-Forwards: 70
Route: <sip:7818888-rrbgth3ot667c@192.168.200.232:5060;lr>
Content-Type: application/sdp
Content-Length:  141

v=0^M
o=user1 53655765 2353687637 IN IP4 192.168.200.231
s=-
c=IN IP4 192.168.200.231
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Outgoing Request INVITE

For outgoing Request from access side, the Register Cache entry will be found by reg-via-key, when options ‘reg-via-key’ and ‘reg-via-match’ are set, the Oracle USM will have wildcarded PAU checking for allow-anonymous (or registered) verification if option wildcard-puid-match is set.

```
INVITE sip:service@192.168.1.232:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.231:5060;branch=z9hG4bK-11911-1-0
From: sipp <sip:7816666@hxu.com>;tag=1
To: sut <sip:service@192.168.1.232:5060>
Call-ID: 1-11911@192.168.1.231
Supported: 100rel,timer,resource-priority,replaces
CSeq: 1 INVITE
Contact: sip:781666@192.168.1.231:5060
Max-Forwards: 70
Content-Type: application/sdp
Content-Length:  137

v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.231
s=-
```

```
c=IN IP4 192.168.1.231
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

IMS Support for Private Header Extensions for 3GPP

As part of its RFC 3455 support, the Oracle USM supports the following headers in its IMS implementation:

- P-Associated-URI
- P-Asserted-Identity
- P-Called-Party-ID
- P-Visited-Network-ID

The procedure to enable IMS access-side support is explained in the access interface configuration section. Any additional configuration required for specific header support is provided below.

P-Associated-URI Header

In the SIP registration process, the registrar often returns a set of associated URIs for a registering AoR. When the Oracle USM receives the list of associated URIs, it stores them in the registration entry for the registering endpoint. The service provider allocates one or more associated URIs per user for his or her own usage. After an endpoint successfully registers, the P-Associated-URI header returned in a 200 OK message informs the UE of all URIs associated with the AoR.

When the Oracle USM receives a request from a UE, the URI in the From header is matched against the registration cache for that endpoint. If the registering endpoint matches an associated-URI already in the registration table, the Service-Route associated with this endpoint is used to create the route for originating transactions associated with the endpoint and the Oracle USM.

The inclusion or exclusion of the P-Associated-URI header is not dependent on the trust level of an ingress or egress realm.

P-Asserted-Identity Header

The Oracle USM inserts a P-Asserted-Identity header into any initial request for a dialog or standalone transaction sourced by the UE.

The inclusion or exclusion of the P-Asserted-Identity header is dependent on the trust level of an egress realm.

P-Asserted-Identity Header Handling

1. The Oracle USM inserts a P-Asserted-Identity header into all messages other than the REGISTER message.
2. When the P-Preferred-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is also present in the UE's associated URI list, then this SIP URI is inserted in the P-Asserted-Identity header as the SIP message enters the core network.

3. When the P-Asserted-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is also present in the UE's associated URI list, then the original P-Asserted-Identity header and SIP URI is passed unchanged into the core network.
4. When the From header is present in an INVITE sourced by the UE, and the SIP URI contained in this header appears in the UE's Associated URI list, then this SIP URI is inserted into the P-Asserted-Identity header as the SIP message enters the core network.
5. When the P-Asserted-Identity header is present in an INVITE sourced by the UE, and the SIP URI contained in this header is not present in the Associated URI list, the Oracle USM acts like no P-Asserted-Identity was received from the UE.
6. When no P-Asserted-Identity can be derived from an INVITE sourced by the UE, the P-Asserted-Identity is based on the first URI in the Associated URI list.
7. The P-Asserted-Identity header will be removed from SIP messages sent and received from a UE if either the ingress or egress side is untrusted and the UE's Privacy header's contents is id.
8. If no P-Associated-URI exists for a registered endpoint, the Oracle USM will use the configured default P-Asserted-Identity found on the sourcing session agent. This feature works with SIP session agents.
9. If the session agent that originates a message does not include a P-Asserted-Identity header or the request is not originated from the session agent, and the Oracle USM has not received P-Associated-URI list from the registrar for a particular user, no P-Asserted-Identity will be created.
10. The P-Preferred-Identity header will never be passed to the Oracle USM's S-CSCF function.

    If the above steps fail to insert a P-Asserted-Identity header, you can manually configure a value to be inserted into a P-Asserted-Identity header. The ims-access parameter must be enabled on the access SIP interface to use the P-Asserted-Identity header override.

P-Asserted-Identity Header Configuration

This section explains how to configure the P-Asserted-Identity header for a session agent using the ACLI or Oracle Communications Session Delivery Manager.

P-Asserted-Identity header handling is enabled with the **ims-access** command on the SIP Interface, as described in the previous section. A P-Asserted-Identity header can be manually configured for a session agent if the automatic logic, explained earlier in this section, fails.

To configure the P-Asserted-Identity header for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Type **select** and the number of the pre-configured session agent you want to configure.

```
ORACLE(session-agent)# select 1
```

5. Type **p-asserted-id** <space> <URI to use when no P-Asserted-ID has been created> and press Enter. This completes the configuration.

   ```
   ORACLE(session-agent)# p-asserted-id sip:name@acmepacket.com
   ```

6. Save your work using the ACLI **done** command.

P-Called-Party-ID Header

The Oracle USM transparently passes the P-Called-Party-ID header between the S-CSCF and a UA.

P-Early-Media SIP Header Support

Version S-CZ7.2.0 provides support for the SIP P-Early-Media that can be used in SIP INVITES, PRACKS, and UPDATES to request and authorize the use of early media. It offers an alternative to policy-based early media support.

The P-Early-Media SIP header is defined in RFC 5009, Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media. RFC 5009 defines the use of the P-Early-Media header within SIP messages in certain SIP networks to authorize the cut-through of backward (server-to-client) and/or forward (client-to-server) early media when permitted by the early media policies of the networks involved. The P-Early-Media header field is intended for use in a SIP network, such as a 3GPP IMS that has the following characteristics: its early media policy prohibits the exchange of early media between end users; it is interconnected with other SIP networks that have unknown, untrusted, or different policies regarding early media; and it has the capability to gate (enable/disable) the flow of early media to/from user equipment.

P-Early-Media SIP Header

The P-Early-Media SIP header is defined in RFC 5009, Private Header (P-Header) Extension to the Session Initiation Protocol (SIP) for Authorization of Early Media. RFC 5009 defines the use of the P-Early-Media header within SIP messages in certain SIP networks to authorize the cut-through of backward (server-to-client) and/or forward (client-to-server) early media when permitted by the early media policies of the networks involved. The P-Early-Media header field is intended for use in a SIP network, such as a 3GPP IMS that has the following characteristics: its early media policy prohibits the exchange of early media between end users; it is interconnected with other SIP networks that have unknown, untrusted, or different policies regarding early media; and it has the capability to gate (enable/disable) the flow of early media to/from user equipment.

A SIP network containing both PSTN gateways and SIP end devices, for example, can maintain such an early media policy by gating "off" any early media with a SIP end device acting as UAS, gating "on" early media with a SIP end device acting as UAC, and gating "on" early media at each PSTN gateway. This is what we have been doing for years with the SIP early media suppression feature, which allows determining who can send early media and in what direction.

Unfortunately, in SIP interconnection scenarios there is no means of assuring that the interconnected network is implementing a compatible early media policy, thus allowing the exchange of user data within early media under some circumstances. For example, if a network "A" allows all early media with user equipment as UAC and an interconnected network "B" allows all early media with user equipment as UAS, any session established between user

equipment as UAC in "A" and user equipment as UAS in "B" will allow bidirectional user data exchange as early media.

The P-Early-Media header is used for the purpose of requesting and authorizing requests for backward and/or forward early media. It's sent from UAS to UAC to indicate authorization for early media.

P-Early-Media-Header Usage

The syntax of the P-Early-Media header field is as follows.

```
P-Early-Media = "P-Early-Media" HCOLON[ em-param *(COMMA em-param) ]
       em-param  = "sendrecv" / "sendonly" / "recvonly" / "inactive" /
"gated" /
      "supported" / token
```

The P-Early-Media header is used for requesting and authorizing requests for backward and/or forward early media. The P-Early-Media header field in an INVITE request contains the "supported" parameter. If P-CSCF is part of the trusted domain, then it must decide whether to insert or delete the P-Early-Media header field before forwarding the INVITE. The P-CSCF upon receiving the P-Early-Media header field in a message towards the UAC needs to verify that the early media request comes from an authorized source. If a P-Early-Media header field arrives from either an untrusted source, a source not allowed to send backward early media, or a source not allowed to receive forward early media, then it may remove the P-Early-Media header field or alter the direction parameter(s) of the P-Early-Media header field before forwarding the message, based on local policy.

The P-Early-Media header field with the "supported" parameter in an INVITE request indicates that the P-CSCF on the path recognizes the header field. The P-Early-Media header field includes one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or "inactive", following the convention used for SDP stream directionality. Each parameter applies, in order, to the media lines in the corresponding SDP messages establishing session media. The parameter value "sendrecv" indicates a request for authorization of early media associated with the corresponding media line, both from the UAS towards the UAC and from the UAC towards the UAS. The value "sendonly" indicates request for authorization of early media from the sender to the receiver and not in the other direction. The value "recvonly" indicates a request for authorization of early media from the receiver, and not in the other direction. The value "inactive" indicates either a request that no early media associated with the corresponding media line be authorized, or a request for revocation of authorization of previously authorized early media. Each parameter applies, in order, to the media lines in the corresponding SDP lines. Unrecognized parameters are discarded and non-direction parameters are ignored. If there are more direction parameters than media lines, the excess are silently discarded. If there are fewer direction parameters than media lines, the value of the last direction parameter applies to all remaining media lines. The P-Early-Media header field in any message within a dialog towards the sender of the INVITE request can also include the non-direction parameter "gated" to indicate that a network entity on the path towards the UAS is already gating the early media, according to the direction parameter(s).

As defined in 3GPP TS 24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3 both the P-CSCF and IBCF may add, remove, or modify, the P-Early-Media header field within forwarded SIP requests and responses according to procedures in RFC 5009.

The P-CSCF can use the P-Early-Media header field for the gate control procedures, as described in 3GPP TS 29.214. Prior to Version S-CZ7.2.0, this capability was based on policy configuration, not examination of the P-Early-Media header.

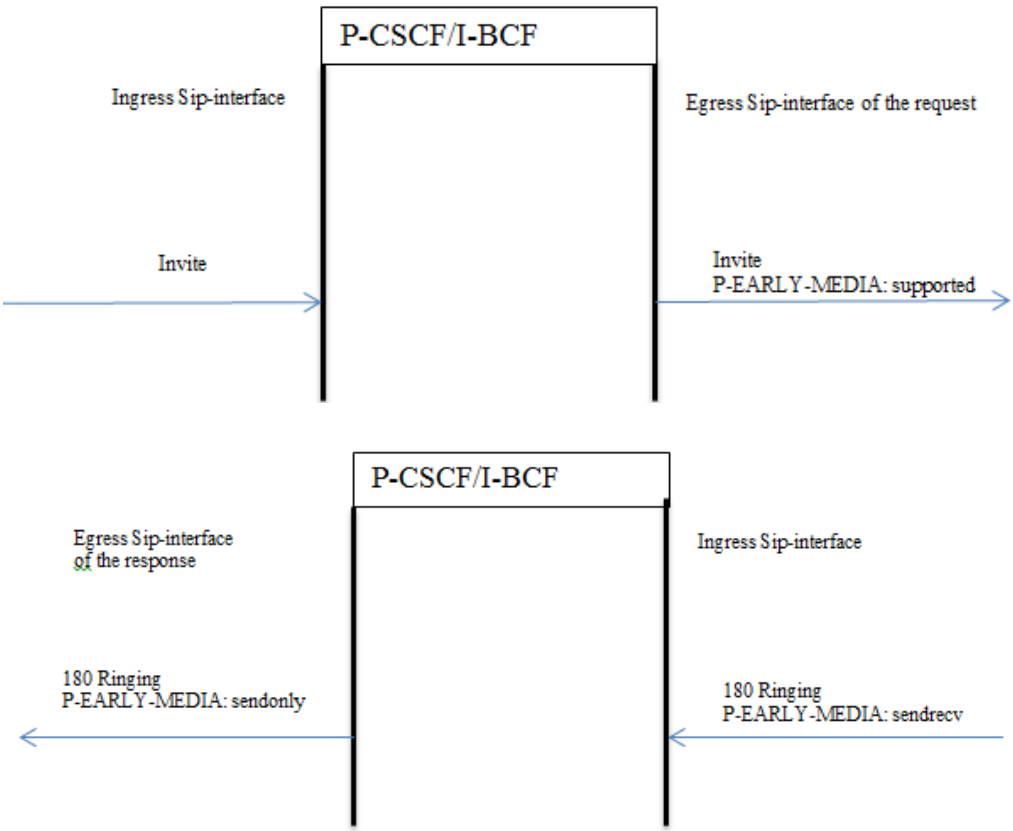
In the current implementation, if the configuration option “early-media-allow” is set to none, the Oracle USM will send the Flow-Status AVP in any AAR request set to disable until the final response.

Functional Design

Acceptance of and authorization for early media is accomplished with two new ACLI parameters -- **p-early-media-header** and **p-early-media-direction**, which are added to SIP interface configuration in Version S-CZ7.2.0 and later releases.

The **p-early-media-header** parameter will enable the feature when the value is set to either “add” or “modify”. The **p-early-media-header** and **p-early-media-direction** should be configured on egress interface of the incoming message. The values for parameter **p-early-media-direction** are “sendrecv, sendonly, recvonly, inactive”. It is a list and each configured value corresponds to the m-line in the SDP. If the number of configured values is more than the number of m-lines in the SDP, the excess configured values are ignored. If the number of configured value is less than the number of m-lines in the SDP, the last configured value is used for all the m-lines.

The following illustrations show the ingress and egress sip-interface configuration.



P-Early-Media headers in Re-Invites are ignored. If the SDP contains Content-Disposition: early-session the P-Early-Media header is ignored.

Endpoint is considered trusted or untrusted based on the configuration on the ingress sip-interface of the P-CSCF. Sip-interface has the configuration parameter “trust-mode”. If the “trust-mode” is set to “none” then nobody is trusted in sip-interface. By default the value is “all”. Possible values are <all, agents-only, realm-prefix, registered, none>.

For multiple dialogs due to forking, P-CSCF will identify the media associated with a dialog, and then setup early media flow for the selected media. The configuration elements restricted-latching in realm-config, and latching in media-router should be enabled.

The following 3 tables detail early media implementation details.

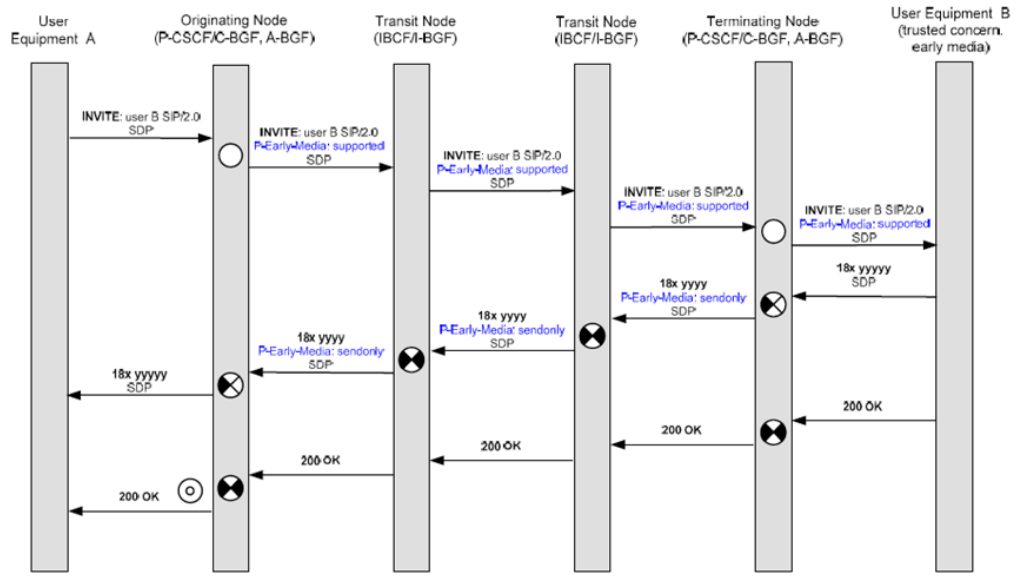
P-Early-Media Trusted/Untrusted to Trusted

This table illustrates the P-CSCF case when messages are received from trusted/untrusted endpoints and forwarded to trusted endpoints.

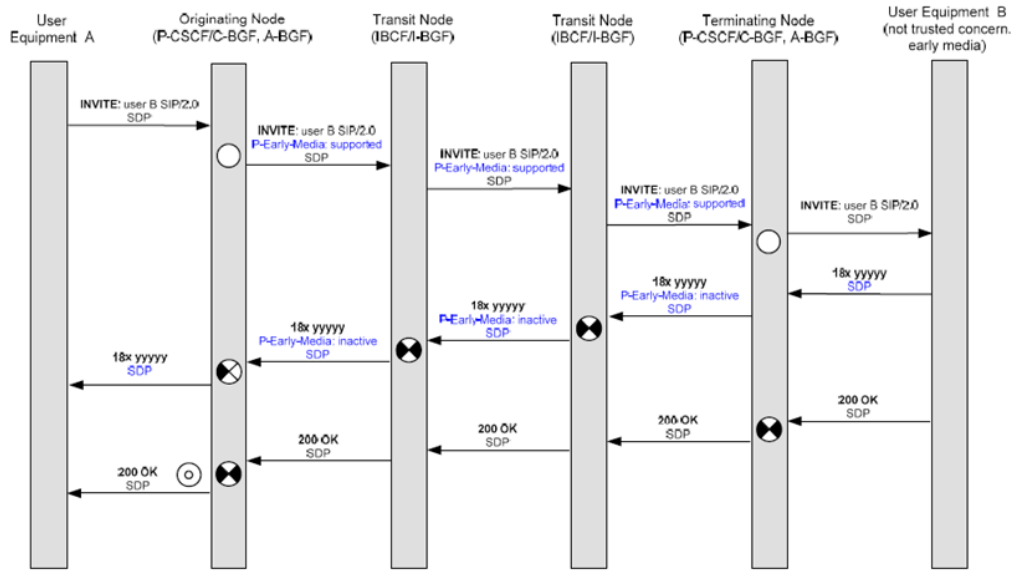
Message -->	INVITE (no PEM)	INVITE (with PEM)	18x (no PEM)	18x (with PEM)	UPDATE/ PRACK (no PEM)	UPDATE/ PRACK (with PEM)
Parameters configured on| egress interface		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"
p-early-media-header-“add” p-early-media-direction - “sendonly	Add header with "supported" value. Setup flows based on configured value.	Add "supported" value if not present. Setup flows based on the value in the header	Add header with configured value and "gated" value.	Add the "gated" value. Setup the flows based on the value in the header.	Add header with configured value and "gated" value. Setup flows based on configured value.	Add the "gated" value . Setup the flows based on the value in the header
p-early-media-header-"modify" p-early-media-direction -"sendonly"	Pass the message transparently.	Add "supported" value if not present. Setup flows based on the value in the header.	Pass the message transparently.	Modify header with configured value and add "gated" value. Setup flows based on configured value.	Pass the message transparently.	Modify header with configured value and add "gated" values. Setup flows based on configured value.
p-early-media-header-"add" No p-early-media-direction	Add header with "supported" value. Setup	Add "supported" value if not present.	Add header with default value and "gated"	Pass the message transparently.	Add header with default value and "gated"	Pass the message transparently.

	flows based on default value.	Setup flows based on the value in the header.	value. Setup flows based on default value.		value. Setup flows based on default value.		
p-early-media-header-"modify" No p-early-media-direction	Pass the message transparently.	Add "supported" value if not present. Setup flows based on the value in the header.	Pass the message transparently.	Modify header with default value and "gated" values. Setup flows based on configured value.	Pass the message transparently.	Modify header with configured value and add "gated" values. Setup flows based on configured value.	Discard header

This illustration shows P-Early-Media set-up with a trusted endpoint.



This illustration shows P-Early-Media set-up with an untrusted endpoint.



P-Early-Media Untrusted to Trusted

This table illustrates the P-CSCF case when messages are received from untrusted endpoints and forwarded to trusted endpoints.

Message -->	INVITE (no PEM)	INVITE (with PEM)	18x (no PEM)	18x (with PEM)	UPDATE/ PRACK (no PEM)	UPDATE/ PRACK (with PEM)	IN /U PR
Parameters configured on| egress interface		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"		with one or more direction parameters where each has one of the values: "sendrecv", "sendonly", "recvonly", or"inactive"	wi un PE
p-early-media-header-“add” p-early-media-direction - “sendonly	Add header with "supported" value. Setup flows based on configured value.	Discard the header.	Add header with configured value and "gated" value. Setup the flows based on configured value.	Discard the header.	Add header with configured value and "gated" value. Setup flows based on configured value.	Discard the header.	Di hea .

p-early-media-header-"modify" p-early-media-direction -"sendonly"	Pass the message transparently.	Discard the header.	Pass the message transparently.	Discard the header.	Pass the message transparently.	Discard the header.	Discard th header.
p-early-media-header-"add" No p-early-media-direction	Add header with "supported" value. Setup flows based on default value.	Discard the header.	Add header with default value and "gated" value. Setup flows based on default value.	Discard the header.	Add header with default value and "gated" value. Setup flows based on default value.	Discard the header.	Discard th header.
p-early-media-header-"modify" No p-early-media-direction	Pass the message transparently.	Discard the header.	Pass the message transparently.	Discard the header.	Pass the message transparently.	Discard the header.	Discard th header.

P-Early-Media Trusted to Untrusted

This table illustrates the P-CSCF case when messages are received from trusted endpoints and forwarded to untrusted endpoints.

Message -->	INVITE (no PEM)	INVITE (with PEM)	18x/ UPDATE/ PRACK (no PEM)	18x (with PEM)	UPDATE/ PRACK (no PEM)	INVITE/18x/ UPDATE/ PRACK
Parameters configured on egress interface						
p-early-media-header-“add” p-early-media-direction - “sendonly	Pass the message transparently.	Delete the header from the message.	Pass the message transparently	Delete the header from the message.	Delete the header from the message.	N/A
p-early-media-header-"modify" p-early-media-direction -"sendonly"	Pass the message transparently.	Delete the header from the message.	Pass the message transparently.	Delete the header from the message.	Delete the header from the message.	N/A

p-early-media-header-"add" No early-media-direction	Pass the message transparently.	Delete the header from the message.	Pass the message transparently	Delete the header from the message.	Delete the header from the message.	N/A
p-early-media-header-"modify" No early-media-direction	Pass the message transparently.	Delete the header from the message.	Pass the message transparently.	Delete the header from the message.	Delete the header from the message.	N/A

P-Early-Media ACLI Configuration

Use the following procedure to configure P-Early-Media SIP header support.

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. Use the **p-early-media-header** parameter to enable P-Early-Media SIP header support. This parameter is disabled by default.
   - **disabled**—(the default value) disables support
   - **add**—enables support and allows the Oracle USM to add the P-Early-Media header to SIP messages.
   - **modify**—enables support and allows the Oracle USM to modify or strip the P-Early-Media header in SIP messages.

```
ACMEPACKET(sip-interface)# p-early-media-header add
ACMEPACKET(sip-interface)#
```

4. Use the **p-early-media-direction** parameter to specify the supported directionalities.
   - **sendrecv**—send and accept early media
   - **sendonly**—send early media
   - **recvonly**—receive early media
   - **inactive**—reject/cancel early media

```
ACMEPACKET(sip-interface)# p-early-media-direction sendrecv,sendrecv
ACMEPACKET(sip-interface)#
```

5. Type **done** to save your configuration.

P-Visited-Network-ID Header

The Oracle USM's IMS support also includes the insertion of a P-Visited-Network-ID header into SIP messages when applicable. When a UE sends a dialog-initiating request (e.g., REGISTER or INVITE message) or a standalone request outside of a dialog (e.g., OPTIONS) to the Oracle USM, it inserts the P-Visited-Network-ID header into the SIP message as it enters into the destination network.

The P-Visited-Network ID header will be stripped from SIP messages forwarded into untrusted networks as expected. The content of a P-Visited-Network-ID header is a text string that identifies the originating UE's home network. This string is user-configurable.

P-Visited-Network-ID Header Handling for SIP Interfaces Configuration

The actual P-Visited-Network-ID string must be configured on the access-side SIP interface. The Oracle USM must consider the egress device trusted or it does not add that the P-Visited-Network-ID header to the forwarded request.

To configure the P-Visited-Network-ID string in a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(sip-interface)# select 1
   ```

5. Type **network-id** <space> <network ID string> and press Enter. This completes the configuration of the P-Visited-Network-ID string for a given SIP interface.

   ```
   ORACLE(sip-interface)# network-id examplenetworkid
   ```

6. Save your work using the ACLI **done** command.

Second P-Asserted-Identity Header for Emergency Calls

The Oracle USM can add a second P-Asserted-Identity header when forwarding an emergency message into the network.

When the UE registers with an S-CSCF, the S-CSCF returns a set of associated URIs, the implicit registration set (IRS,) for the AoR in the 200 OK response. The Oracle USM caches the IRS. The user identities that comprise the cached IRS are used for validation later.

As the Oracle USM receive a UE's INVITE, the value in the P-Preferred-Identity header is validated against the public user identities in the cached IRS. If the inbound P-Preferred-Identity matches any items in the IRS, the Oracle USM inserts that value into a P-Asserted-Identity header in the egress message. The P-Preferred- Identity header is stripped from the outbound message.

Inbound INVITE Contains:	**Validate Against Implicit Registration Set**	**Outbound Invite Created With: (all P-Preferred-Identity headers are removed)**
P-Preferred-Identity: value-1	value-1 present	P-Asserted-Identity: value-1

The Oracle USM can create a second P-Asserted-Identity header by configuring the add-second-pai option in the SIP config. Once a combination of SIP URIs and/or TEL URIs in the inbound P-Preferred-Identity header are validated against the IRS, the Oracle USM forwards the emergency INVITEs with two P-Asserted-Identity headers to the E-CSCF. The behavior is based upon the URI type received in the P-Preferred-Identity header and whether those values validate against the IRS list.

Inbound INVITE Contains:	**Validate Against Implicit Registration Set**	**Outbound Invite Created With: (all P-Preferred-Identity headers are removed)**
P-Preferred-Identity: SIP-URI-1	SIP-URI-1 present TEL-URI-1 present (TEL-URI-n present)	P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1
P-Preferred-Identity: TEL-URI-1	TEL-URI-1 present SIP-URI-1 present (SIP-URI-n present)	P-Asserted-Identity: TEL-URI-1 P-Asserted-Identity: SIP-URI-1
NO P-Preferred-Identity:	(SIP | TEL) URI -1 present (SIP | TEL) URI -n present	P-Asserted-Identity: 1st public identity from IRS
P-Preferred-Identity: SIP-URI-1	TEL-URI-1 present	P-Asserted-ID: SIP-URI-1
P-Preferred-Identity: TEL-URI-1	SIP-URI-1 present	P-Asserted-ID: TEL-URI-1
P-Preferred-Identity: SIP-URI-1	SIP-URI-1 present	P-Asserted-Identity: SIP-URI-1
P-Preferred-Identity: SIP-URI-2	SIP-URI-2 present TEL-URI-n present	P-Asserted-Identity: 1st TEL-URI from IRS
P-Preferred-Identity: TEL-URI-1	TEL-URI-1 present	P-Asserted-Identity: TEL-URI-1
P-Preferred-Identity: TEL-URI-2	TEL-URI-2 present SIP-URI-n present	P-Asserted-Identity: 1st SIP-URI from IRS

If the INVITE does not include a P-Preferred-Identity header and does not include a P-Asserted-Identity header, or the value in the original P-Preferred-Identity or P-Asserted-Identity header is not contained in the IRS, or the URI from the From: header is not the in the IRS, then default public user identity is inserted into a P-Asserted- Identity header in the egress message. (The default public user identity is the first on the list of URIs in the P-Associated-URI header.)

If the strict-3gpp-pai-compliance option is configured in the outbound SIP interface, the first P-Asserted-Identity header also includes the display name.

Two Incoming P-Asserted-Identity Headers

When the inbound INVITE contains 2 P-Asserted-Identity headers, the Oracle USM ensures that both outbound P-Asserted-Identity headers contain public user identities from the IRS according to the following:

Inbound Invite Contains	Validate Against Implicit Registration Set:	Outbound Invite Created With:
P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1	SIP-URI-1 and TEL-URI-1 present	P-Asserted-Identity: SIP-URI-1 or TEL-URI-1 P-Asserted-Identity: 1st public identity from IRS other than value in 1st P-Asserted-Identity
P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: SIP-URI-2	SIP-URI-1 or SIP-URI-2 present TEL-URI-1 present	P-Asserted-Identity: SIP-URI-1 P-Asserted-Identity: TEL-URI-1
P-Asserted-Identity: TEL-URI-1 P-Asserted-Identity: TEL-URI-2	TEL-URI-1 or TEL-URI-2 present SIP-URI-1 present	P-Asserted-Identity: TEL-URI-1 or TEL-URI-2 P-Asserted-Identity: SIP-URI-1

1. Navigate to the sip-config configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. Type select to begin configuring this object.

```
ACMEPACKET(sip-config)# select
ACMEPACKET(sip-config)#
```

3. options—Configure the add-second-pai option:

```
ACMEPACKET(sip-config)# options +add-second-pai
ACMEPACKET(sip-config)#
```

4. Save and activate your configuration.

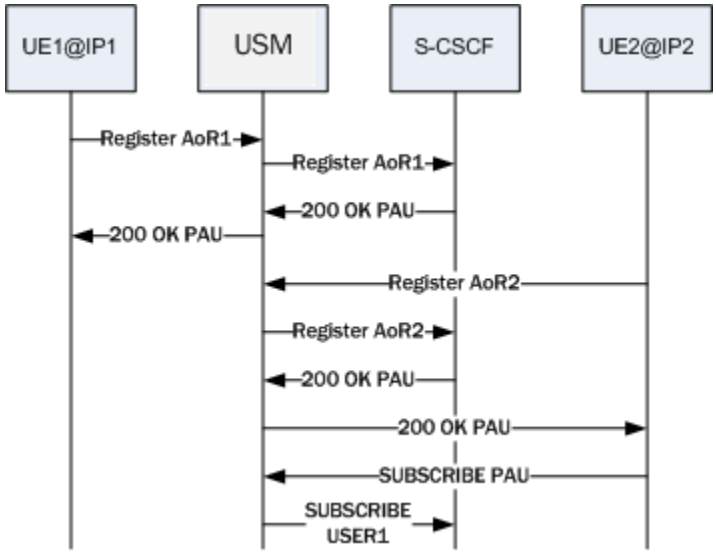
Temporary Public User Identities and Multi-SIM Scenarios

The Oracle USM's SIP interface supports multiple registered users for the same P-Asserted-Uri (PAU), a useful ability for multi-SIM scenarios. The call flow for this type of scenario differs depending on whether or not you configure the SIP interface facing the UE with the **reg-via-key** and **reg-via-match** options.

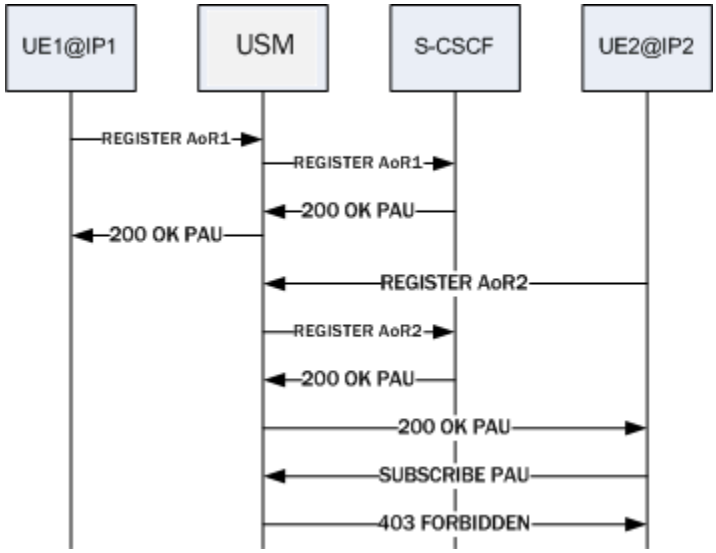
In a multi-SIM scenario, the UE derives a temporary IMS public identity (IMPU); that UE then registers with the Oracle USM using the IMPU as the address of record (AoR) from a unique IP. The S-CSCF returns a PAU in the 200 OK, which the Oracle USM caches. The UE then derives another IMPU; it registers with the Oracle USM using that IMPU as the AoR from another unique IP. The S-CSCF again returns the same PAU in the 200OK.

Old Behavior

Before the introduction of this change, the Oracle USM associated the PAU only with the first IMPU request. The Oracle USM considered any request made from that PAU to be a request from the first user, regardless of the request's originating IP.

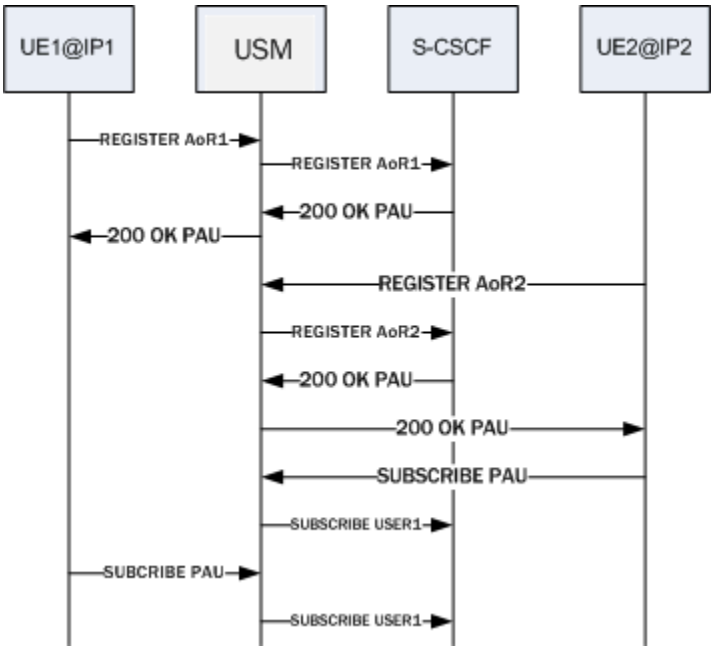


When the SIP interface facing the UE had the **reg-via-key** and **reg-via-match** options configured, the Oracle USM rejected the request from the second user with the PAU as the From or the PPI. Because it only associates the PAU with the first user, the Oracle USM issued a 403 message.

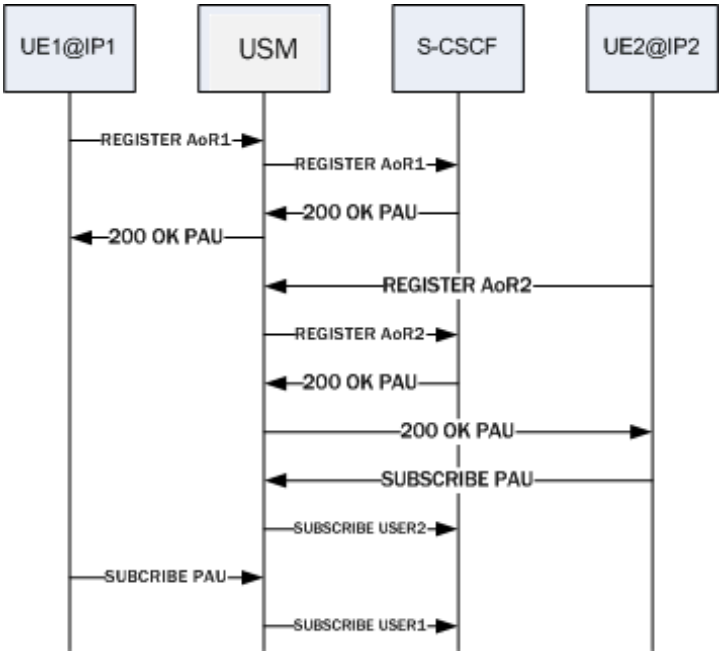


New Behavior

Your Oracle USM now associates the PAU with both the first and second IMPU. The Oracle USM considers any request made from that PAU to be a request from the user at the top of its registration cache table irrespective of where the request originated (the IP).



When the SIP interface facing the UE has the **reg-via-key** and **reg-via-match** options configured, and the request from the user with the PAU as the From or with PPI, the Oracle USMmatches to the proper user based on the source IP.



Configuring SIP Interface with reg-via-key and reg-via-match

If you do not want the use the call scenario associated with the SIP interface options in the New Behavior section, you do not need to make any change to your configuration.

If you want your call scenarios to resemble the one associated with the SIP interface options in the New Behavior section, then you need to configure **reg-via-key** and **reg-via-match** options on the SIP interface facing the UE.

To configure a SIP interface with the reg-via-key and reg-via-match options:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name. Then press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-key
   ORACLE(sip-interface)# options +reg-via-match
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

IMS-AKA

The Oracle USM supports IP Media Subsystem-Authentication and Key Agreement (IMS-AKA).

Defined in 3GPPr7 (specifications in TS 33.203 and call flows in TS 24.228), IMS-AKA can be used as a framework for authentication and for securing the signaling path between a UE and the Oracle USM (when the Oracle USM is acting as a P-CSCF or as a B2BUA) across the Gm interface.

In addition, the Oracle USM's serving as an IMS-AKA termination point is valuable because it allows IMS-AKA use behind by multiple endpoints sitting behind a NAT device. IMS-AKA support also works when there are no NAT devices between endpoints and the Oracle USM acting as a P-CSCF, and when the Oracle USM sits behind a third-party P-CSCF. In addition, you can use IMS-AKA when the endpoint uses SIP UDP.

Requirements

IMS-AKA use assumes that you have installed the appropriate IPSec module on your Oracle USM, or that it has come from Oraclewith those modules pre-installed. IMS-AKA will not work without this hardware.

In addition, your configuration must have SIP registration caching enabled.

The refreshRegForward Option

The Oracle USM provides a the user with a means of ignoring its registration refresh half-life timer, and send all applicable registration refreshes received via IMS-AKA to the core for authentication.

By default, the Oracle USM uses its half-life function and attempts to manage registration refreshes prior to half-life expiry without forwarding the refresh to the core. The Oracle USM sends registration refreshes that arrive after the half-life expiry to the core.

The user changes this behavior by setting the **refreshRegForward** in the applicable IMS-AKA profile to as follows.

```
ORACLE(ims-aka-profile)# options +refreshRegForward
```

When this option is set, the system forwards every refresh registration to the IMS core regardless of the half-life timer's status.

Monitoring

The ACLI **show sipd endpoint-ip** command is updated to show the IMS-AKA parameters corresponding to each endpoint. The display shows the algorithms used, the ports used, and the security parameter indexes (SPIs) used.

In addition, the **show sa stats** command now shows the security associations information for IMS-AKA.

ACLI Instructions and Examples

You enable IMS-AKA by configuring the following:

- An IMS-AKA profile
- Certain parameters in the global IPSec configuration
- Certain parameters in the SIP interface, and in the SIP interface's SIP port

Setting Up an IMS-AKA Profile

An IMS-AKA profile establishes the client and server ports to be protected, and it defines lists of encryption and authentication algorithms the profile supports. You can configure multiple IMS-AKA profiles, which are uniquely identified by their names.

You apply an IMS-AKA profile to a SIP port configuration using the name.

To configure an IMS-AKA profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ims-aka-profile** and press Enter.

```
ORACLE(system)# ims-aka-profile
ORACLE(ims-aka-profile)#
```

4. **name**—Enter the name you want to give this IMS-AKA profile. This is the value you will use to apply the profile to a SIP port configuration. This parameter is required, and it has no default value.
5. **protected-server-port**—Enter the port number of the protected server port, which is the port on which the Oracle USM receives protected messages. The protected server port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.
6. **protected-client-port**—Enter the port number of the protected client port, which is the port on which the Oracle USM sends out protected messages. Like the protected server port, the protected client port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.
7. **encr-alg-list**—Enter the list of encryption algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   ```
   ORACLE(ims-aka-profile)# encr-alg-list "aes-cbc null"
   ```

   This parameter defaults to the following three values: **aes-cbc**, **des-ede3-cbc**, and **null**.
8. **auth-alg-list**—Enter the list of authentication algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   ```
   ORACLE(ims-aka-profile)# auth-alg-list "hmac-sha-1-96 hmac-md5-96"
   ```

   This parameter defaults to hmac-sha-1-96.

Setting Up an IPSec Profile for IMS-AKA Use

Using the global IPSec configuration, you establish the parameters governing system-wide IPSec functions and behavior. This configuration also contains parameters required for IMS-AKA support. The IPSec global configuration is a single instance element, meaning there is one for the whole system.

To configure the global IPSec parameters that apply to IMS-AKA:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```
2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```
3. Type **ipsec** and press Enter.

```
ORACLE(system)# ipsec
ORACLE(ipsec)#
```

4. Type **ipsec-global-config** and press Enter. If you are editing a pre-existing IPsec global configuration, then you need to select the configuration before attempting to edit it.

```
ORACLE(system)# ipsec-global-config
ORACLE(ipsec-global-config)#
```

5. **red-ipsec-port**—Specify the port on which the Oracle USM should listen for redundancy IPSec synchronization messages. The default is 1994, and valid values are in the range from 1025 to 65535.
6. **red-max-trans**—Enter the maximum number of redundancy transactions to retain on the active. The default is 10000, and valid values range up to a 999999999 maximum.
7. **red-sync-start-time**—Enter the time in milliseconds before the system starts to send redundancy synchronization requests. The default is 5000, and valid values range up to a 999999999 maximum.
8. **red-sync-comp-time**—Enter the time in milliseconds to define the timeout for subsequent synchronization requests once redundancy synchronization has completed. The default is 1000, and valid values range up to a 999999999 maximum.

Enabling IMS-AKA Support for a SIP Interface

To enable IMS-AKA for a SIP interface, you must set the **sec-agree-feature** parameter to enabled.

To enable IMS-AKA for a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing SIP interface, you need to select and edit that configuration.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **sec-agree-feature**—Change this parameter to **enabled** if you want to use IMS-AKA on this SIP interface. By default, this parameter is **disabled**.

Applying an IMS-AKA Profile to a SIP Port

The final step in setting up IMS-AKA support is to apply an IMS-AKA profile to a SIP port. Enter the **name** value from the IMS-AKA profile you want to apply in the SIP port's **ims-aka-profile** parameter.

To apply an IMS-AKA profile to a SIP port:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing SIP port, you need to select and edit that configuration.

```
ORACLE(session-interface)# sip-ports
ORACLE(sip-port)#
```

5. **ims-aka-profile**—Enter the **name** value for the IMS-AKA profile configuration you want applied to this SIP port. This parameter has no default.
6. Save and activate your configuration.

IPSec IMS-AKA

Compliance with the VoLTE specification (GSMA PRD IR.92) requires cluster member support for IPsec IMS-AKA (IP Multimedia Services Authentication and Key Agreement) as defined in 3GPP TS 24.299, *IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*: Stage 3, and TS 33.203, *3G Security: Access Security for IP-based Services*.

Support for IMS-AKA requires no new additional configuration elements.

Sample IMS-AKA Configuration

The following formatted extract from **show running-config** ACLI output shows a sample IMS-AKA profile configuration.

```
ims-aka-profile
name                          dut2.test
protected-client-port         4060
protected-server-port         4060
encr-alg-list                 aes-cbc des-ede3-cbc null
auth-alg-list                 hmac-sha-1-96 hmac-md5-96
last-modified-by              admin@172.30.11.18
last-modified-date            2012-01-10 17:31:59
```

Sample Security Policy Configuration

The following formatted extracts from **show running-config** ACLI output shows three associated security policies.

The first policy, and the one with the highest priority, opens Port 5060 for SIP traffic.

```
security-policy
name                          pol1
network-interface             M10:0.6
priority                      0
local-ip-addr-match           3fff:c0ac::c0ac:ce12
remote-ip-addr-match          ::
local-port-match              5060
remote-port-match             0
trans-protocol-match          ALL
direction                     both
```

```
local-ip-mask                   ::
remote-ip-mask                  ::
action                          allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask                 65535
remote-port-mask                65535
trans-protocol-mask             0
valid                           enabled
vlan-mask                       0xFFF
last-modified-by                admin@console
last-modified-date              2012-01-10 17:48:59
```

The second policy opens Port 4444 for CCP traffic.

```
security-policy
name                            pol2
network-interface               M10:0.6
priority                        2
local-ip-addr-match             3fff:b623::b623:ce02
remote-ip-addr-match            3fff:b623::b623:ce01
local-port-match                4444
remote-port-match               4444
trans-protocol-match            ALL
direction                       both
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
action                          allow
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask                 65535
remote-port-mask                65535
trans-protocol-mask             0
valid                           enabled
vlan-mask                       0xFFF
last-modified-by                admin@console
last-modified-date              2012-01-10 17:49:15
```

The third policy, the policy with the least priority, and, consequently, the last policy applied, requires IPsec on all ports.

```
security-policy
name                            pol3
network-interface               M10:0.6
priority                        10
local-ip-addr-match             3fff:c0ac::c0ac:ce12
remote-ip-addr-match            ::
local-port-match                0
remote-port-match               0
trans-protocol-match            ALL
direction                       both
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
remote-ip-mask                  ::
action                          ipsec
ike-sainfo-name
outbound-sa-fine-grained-mask
local-ip-mask                   ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

```
remote-ip-mask                ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
local-port-mask               65535
remote-port-mask              65535
trans-protocol-mask           0
valid                         enabled
vlan-mask                     0xFFF
last-modified-by              admin@console
last-modified-date            2012-01-10 17:50:42
```

Sec-Agree

Version S-CZ7.2.0 introduces support for RFC 3329, Security Mechanism Agreement for the Session Initiation Protocol, commonly referred to as Sec-Agree. The RFC defines three SIP headers, Security-Client, Security-Server, and Security-Verify that provide the ability for SIP UAs and other SIP entities (servers, proxies, and registrars) to negotiate next-hop security mechanisms. Note that this initial implementation does not provide support for server-initiated security negotiation, nor does it support media-plane security. That is, support is limited to client-initiated negotiation during initial registration, and to signalling security.

Currently the P-CSCF functionality includes support for IMS-AKA feature for VoLTE deployments. In order to support RCS clients along with VoLTE P-CSCF functionality needs to be enhanced to support RFC 3329, Security Mechanism Agreement for the Session Initiation Protocol (commonly referred to as Sec-Agree), which includes support for TLS as security mechanism.

Sec-Agree defines three SIP headers, Security-Client, Security-Server and Security-Verify, to negotiate security agreements during initial REGISTER transactions. Header definitions are as follows:

```
security-client  = "Security-Client" HCOLON
                          sec-mechanism *(COMMA sec-mechanism)
security-server  = "Security-Server" HCOLON
                   sec-mechanism *(COMMA sec-mechanism)
security-verify  = "Security-Verify" HCOLON
                   sec-mechanism *(COMMA sec-mechanism)
sec-mechanism    = mechanism-name *(SEMI mech-parameters)
mechanism-name   = ( "digest" / "tls" / "ipsec-ike" /
                     "ipsec-man" / token )
mech-parameters  = ( preference / digest-algorithm /
                     digest-qop / digest-verify / extension )
preference       = "q" EQUAL qvalue
qvalue           = ( "0" [ "." 0*3DIGIT ] )/ ( "1" [ "." 0*3("0") ] )
digest-algorithm = "d-alg" EQUAL token
digest-qop       = "d-qop" EQUAL token
digest-verify    = "d-ver" EQUAL LDQUOT 32LHEX RDQUOT
extension        = generic-param
```

The Security-Client header contains one or more security mechanisms and associated parameters proposed by the initiating client. This initial implementation supports two security mechanisms: TLS and ipsec-3gpp.

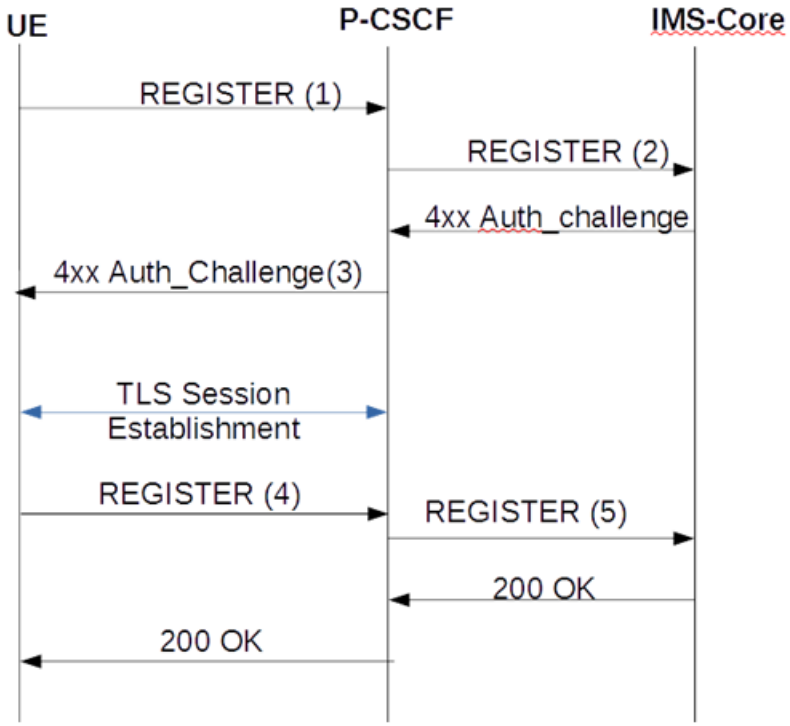
The Security-Server header contains the security mechanism chosen by the server from those mechanisms proposed by the client.

The Security-Verify header contains the contents of the Security-Server header.

Two additional header fields, Require and Proxy-Require, are also used in support of Sec-Agree negotiations. Both headers are required in client transmissions.

TLS Session Setup During Registration

This call flow depicts a TLS session setup during the registration procedure. Only relevant header fields are noted.



(1) REGISTER

```
Proxy-Require:sec-agree
Security-Client: ipsec-3gpp;alg=hmac-md5-96;ealg=aes-cbc;prot=esp;mod=trans;spi-
c=8765423;port-c=7524;spi-s=1234563;port-s=1358, ipsec-3gpp;alg= hmac-
sha-1-96;ealg=aes-cbc;prot=esp;mod=trans;spi-c=8765423;port-c=7524;spi-
s=1234563;port-s=1358, tls
```

(2) REGISTER

```
Authorization: Digest
uri="sip:ims.mnc007.mcc262.3gppnetwork.org",username="262073900320132@ims.mnc007.
mcc262.3gppnetwork.org",response="",realm="ims.mnc007.mcc262.3gppnetwork.org",non
ce=""
(No integrity-protected field will be present if TLS is selected as the security
mechanism)
```

(3) 401

```
Security-Server: tls
```

(4) REGISTER

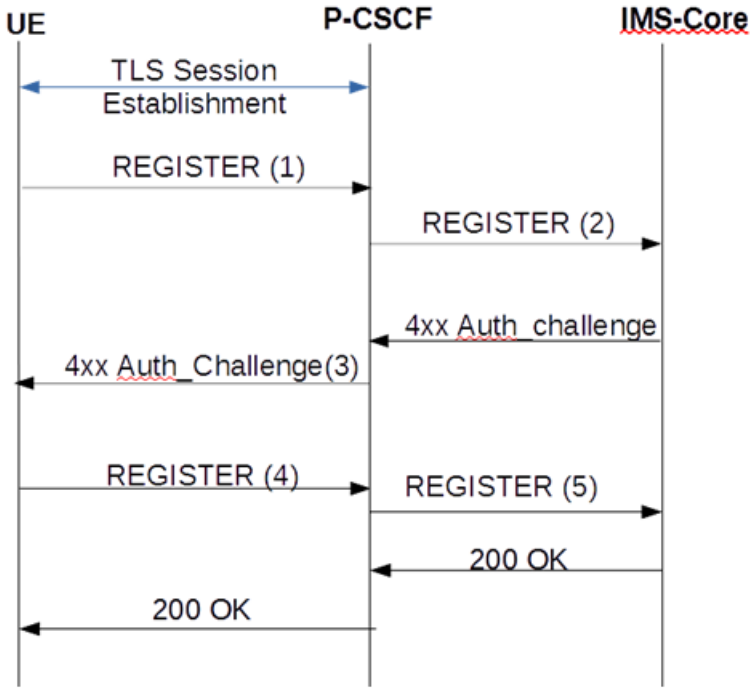
```
Proxy-Require: sec-agree
Security-Verify: tls
```

(5) REGISTER

```
Authorization: Digest
username="262073900320132@ims.mnc007.mcc262.3gppnetwork.org",realm="imstest1.tele
fonica.de",uri="sip:ims.mnc007.mcc262.3gppnetwork.org",qop=auth,nonce="Ms8l2xeF3l
4lb0VO8fK3KFMSKLv1sQAATdN2NpFUCgU=",nc=00000001,cnonce="3063397945",algorithm=AKA
v1-MD5,response="3779ff40a057f999a2f8288bbfafc10d", integrity-protected=tls-
pending
```

TLS Session Setup Prior to Registration

This call flow depicts a TLS session setup prior to the registration procedure.



(1) REGISTER

```
(No Security-Client or Proxy-Require header present)
```

(2) REGISTER

```
Authorization: Digest
uri="sip:ims.mnc007.mcc262.3gppnetwork.org",username="262073900320132@ims.mnc007.
mcc262.3gppnetwork.org",response="",realm="ims.mnc007.mcc262.3gppnetwork.org",non
ce=""
(No integrity-protected field will be present)
```

(3) 401

```
(No Security-Server header present)
```

(4) REGISTER

```
(No Security-Client, or Security-Verify or Proxy-Require header present)
```

(5) REGISTER

```
Authorization: Digest
username="262073900320132@ims.mnc007.mcc262.3gppnetwork.org",realm="imstest1.tele
fonica.de",uri="sip:ims.mnc007.mcc262.3gppnetwork.org",qop=auth,nonce="Ms8l2xeF3l
```

```
4lb0VO8fK3KFMSKLv1sQAATdN2NpFUCgU=",nc=00000001,cnonce="3063397945",algorithm=AKA
v1-MD5,response="3779ff40a057f999a2f8288bbfafc10d", integrity-protected=tls-
pending
```

Regardless of TLS Session setup procedure, if the newly added configurable item sec-agree feature is enabled, any messages on unprotected port will be rejected except REGISTER messages or messages related to emergency services.

For refresh registration, if the Sec_Agree occurred during Registration, it verifies for the presence or change of Security-Client & Security-Verify headers, if they differ it will be rejected with 4xx response and also Authorization header fields are verified irrespective of the above methods and if they differ with previous association it will be rejected with 403 (Forbidden) response. Also when the refresh REGISTER is being forward to the core, it will set the integrity-protected field to "tls-yes".

SEC-agree Configuration

1. Access the **sip-interface** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

2. Select the **sip-interface** object to edit.

   ```
   ORACLE(sip-interface)# select
   <RealmID>:
   1: realm01 172.172.30.31:5060

   selection: 1
   ORACLE(sip-interface)#
   ```

3. **sec-agree-feature**—Set this parameter to enable or disable for Sec-Agree support. By default, support is disabled.
4. **sec-agree-pref**—Configure this parameter to specify security protocol preferences.
   - **ipsec3gpp** — support only IMS-AKA protocol
   - **tls** — support only TLS protocol
   - **ipsec3gpp-tls** — support both IMS-AKA and TLS, preferred protocol is IMS-AKA
   - **tls-ipsec3gpp** — support both TLS and IMS-AKA, preferred protocol is TLS
5. Type **done** to save your configuration.

IMS AKA over TCP

IMS-AKA registration is conducted over UDP or TCP protocol only. The Oracle USM supports both transport protocols.

Within mobile IMS VoLTE/RCSe deployments, IP packets carrying SIP messages can be large due to IPv6 headers, IMS-AKA specific headers, extensive codec policies, and other 3GPP related headers. Because of this, IPv6 VoLTE signaling messages using IMS-AKA frequently exceed 1300 bytes and require TCP according to RFC3261 section 18.1.1.

IMS-AKA Secure Call Registration over TCP

To register and place a call into the network, a UE creates 3 TCP connections. The first insecure connection is established to the port (usually 5060) specified in the sip-port for the first registration request. You should create a sip-ports configuration element with port 5060 and an ims-aka-profile parameter that references an ims-aka-profile configuration element. The ims-aka-profile configuration element initiates the process that creates secure connections. For example:

sip-ports

```
address                       sd-ip-address
port                          5060
transport-protocol            TCP
tls-profile
multi-home-addrs
allow-anonymous               registered
ims-aka-profile               profile
```

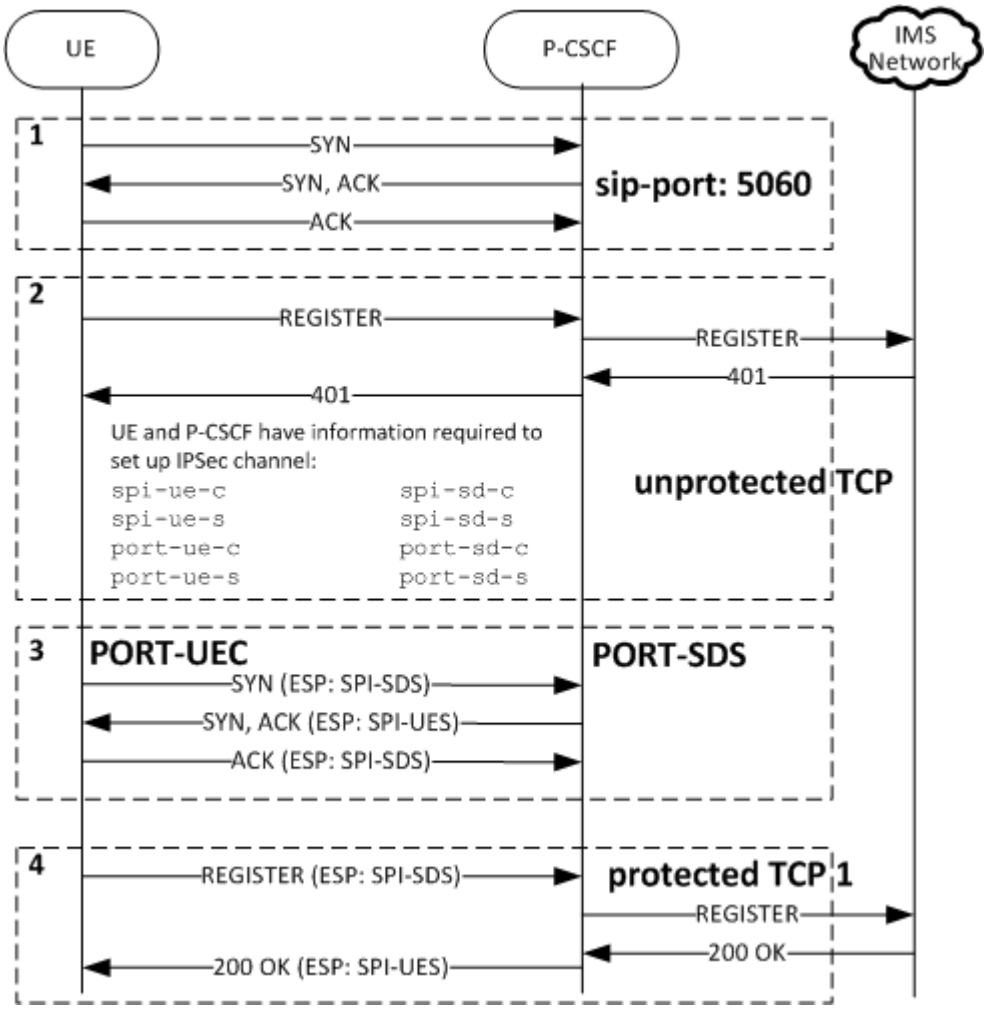
The ims-aka-profile configuration element defines the protected-server-port (PORT-SDS) and protected-client-port (PORT-SDC). The protected-server-port is opened for both inbound TCP and UDP traffic. For example:

ims-aka-profile

```
name                          profile
protected-client-port         PORT-SDC
protected-server-port         PORT-SDS
encr-alg-list                 aes-cbc des-ede3-cbc null
auth-alg-list                 hmac-sha-1-96
```

When the UE receives the 401Unauthorized challenge from the Oracle USM acting as P-CSCF, both devices have the information to set up security association for two IPSec channels. The UE establishes the second TCP connection via IPSec channel from the UE's PORT-UEC to the P-CSCF's PORT-SDS, and the registration process continues.

Hereafter, the UE uses the IPSec channels from communication.

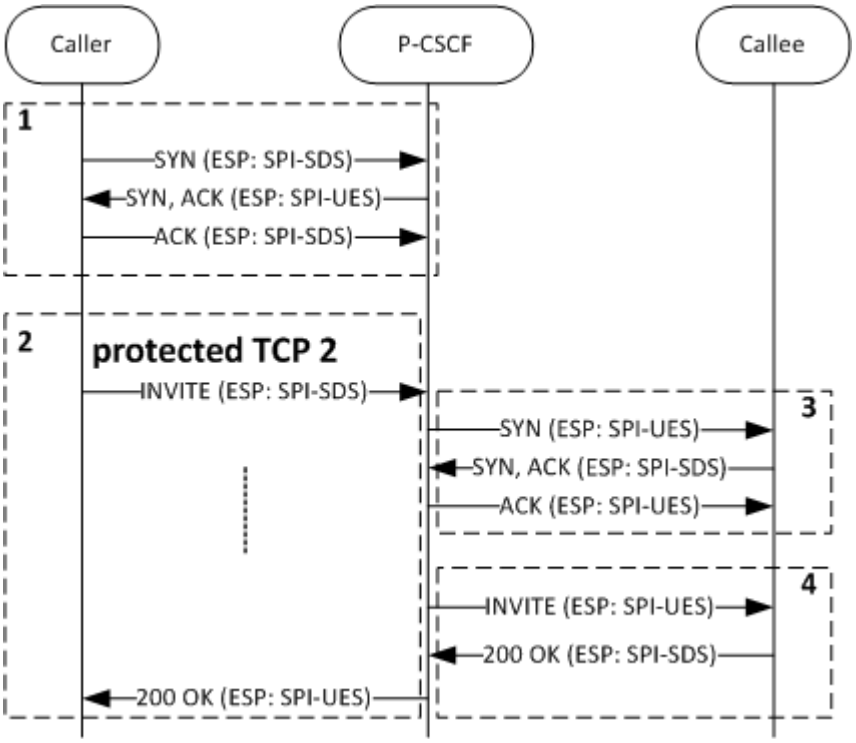


1. The UE and P-CSCF set up the TCP connection.
2. The UE sends an unauthenticated SIP Register message to the P-CSCF's unprotected server port (usually 5060). The Register message is forwarded to the UE's Home S-CSCF. The S-CSCF then replies with a the SIP 401 Authentication Required response back to P-CSCF. This message contains encryption keys and authentication information.

   The P-CSCF modifies the 401 message back to UE. At this point, both UE and P-CSCF should have all the information need to establish secure IPSec channels.
3. The UE and P-CSCF create a TCP connection over a secure channel from port-ue-c to port-sd-s.
4. The UE sends an authenticated REGISTER over the secure channel via the P-CSCF to the S-CSCF. If the authentication is valid, the P-CSCF will forward the 200 OK response from the S-CSCF to the UE. The 200 OK response will be sent in the same secure TCP connection.

IMS-AKA Call Establishment over TCP

For the UE to send a new request into the network and establish a call, a third TCP connection is created using the information configured/generated prior to creating the first secure connection.



1. If the TCP connection over a secure channel does not exist, the Caller will establish it.
2. The Caller initiates the call by sending SIP INVITE from its PORT-UEC to the P-CSCF's PORT-SDS.
3. The P-CSCF forwards the INVITE to the Caller. If not present, it will create a secure TCP connection from its PORT-SDC to the callee's PORT-UES.
4. The INVITE is then forwarded to the Callee securely.

SIP SUBSCRIBE and NOTIFY over TCP IMS-AKA

SUBSCRIBE and NOTIFY messages are exchanged between a UE and the P-CSCF in a manner similar to the previous INVITE example whereby the secure channel is first created and then the SIP messages are exchanged securely.

IMS-AKA Change Client Port

The Oracle USM is now in compliance with 3GPP TS 33.203, *Access Security for IP-Based Services*. Previous releases did not comply with requirements specified in Section 7.4, Authenticated re-registration, which reads in part:

Every registration that includes a user authentication attempt produces new security associations. If the authentication is successful, then these new security associations shall replace the previous ones. This clause describes how the UE and P CSCF handle this replacement and which SAs to apply to which message.

When security associations are changed in an authenticated re-registration then the protected server ports at the UE (port_us) and the P-CSCF (port_ps) shall remain unchanged, while the protected client ports at the UE (port_uc) and the P-CSCF (port_pc) shall change.

If the UE has an already active pair of security associations, then it shall use this to protect the REGISTER message. If the S-CSCF is notified by the P-CSCF that the REGISTER message from the UE was integrity-protected it may decide not to authenticate the user by means of the AKA protocol. However, the UE may send unprotected REGISTER messages at any time. In this case, the S-CSCF shall authenticate the user by means of the AKA protocol. In particular, if the UE considers the SAs no longer active at the P-CSCF, e.g., after receiving no response to several protected messages, then the UE should send an unprotected REGISTER message."

Prior releases failed to change the protected client ports after a successful re-registration.

Protected Ports

Within IMS networks, the P-CSCF provides the network access point and serves as the outbound proxy server for user equipment -- smart phones, tablets, and similar devices. The UE must connect to the P-CSCF prior to registration and initiation of SIP sessions. Connection to the P-CSCF, which can be in the user's home network, or in a visited network if the UE is roaming, is accomplished using Dynamic Host Control Protocol (DHCP) P-CSCF discovery procedures.

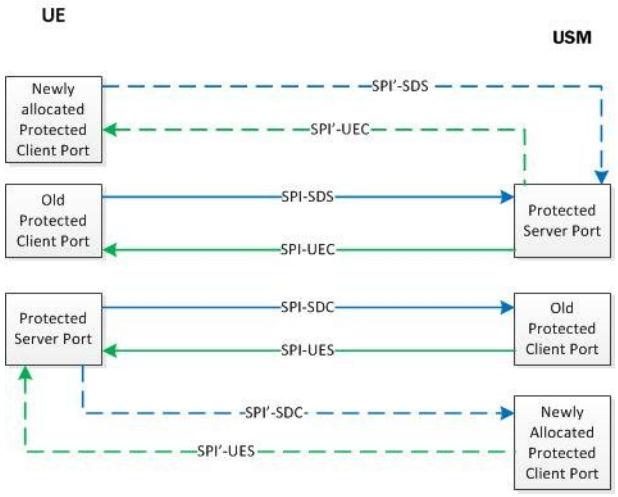
After successful discovery, the P-CSCF and UE negotiate IPSec security associations (SAs) which are used to establish four protected (authenticated and encrypted using Encapsulating Security Payload protocol) ports between the UE and the P-CSCF.

The four protected ports are shown in the following illustration:

As required by Section 7.4 of 3GPP TS 33.203, the protected client ports, one on the UE and the other on the Oracle USM, must be changed after each successful re-registration.

To fulfill this requirement, this release adds a new attribute to the existing ims-aka-profile configuration object. This attribute (**end-protected-client-port**) works in conjunction with **start-protected-client-port** (**protected-client-port** in previous releases) to enable the identification of a pool of protected client ports, which will be used for re-registration scenarios where the Oracle USM is required to change the client port.

The Oracle USM creates new protected client ports, one on the UE and the other on the Oracle USM, after every re-registration. Old protected client ports, along with their associated SAs, are maintained for 30 seconds after re-registration to ensure correct handling or any pending responses to previously transmitted messages.



After successful re-registration, the Oracle USM updates the registration cache with updated port information and checkpoint with the HA peer, if present.

IMS-AKA Change Client Port Configuration

An IMS-AKA profile establishes the client and server ports to be protected, and it defines lists of encryption and authentication algorithms the profile supports. You can configure multiple IMS-AKA profiles, which are uniquely identified by their names.

You apply an IMS-AKA profile to a SIP port configuration using the name.

To configure an IMS-AKA profile:

1. From Superuser mode, use the following command sequence to navigate to ims-aka-profile configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ims-aka-profile
   ORACLE(ims-aka-profile)#
   ```

2. **name**—Enter the name you want to give this IMS-AKA profile. This is the value you will use to apply the profile to a SIP port configuration. This parameter is required, and it has no default value.

3. **protected-server-port**—Enter the port number of the protected server port, which is the port on which the Oracle USM receives protected messages. The protected server port should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

4. **start-protected-client-port (protected-client-port in Release S-CX6.3.3M2 and earlier releases)**—Enter the start value for the pool of port numbers available following a successful re-authentication. Like the protected server port, the protected client port pool should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the

steering port used in a call will prevent SIP messages from reaching the system's host processor.

Any existing configuration for **protected-client-port** will be mapped to both **start-protected-client-port** and **end-protected-client-port** parameter values.

This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

5. **end-protected-client-port**—Enter the end value for the pool of port numbers available following a successful re-authentication. Ensure that this value is greater than the value assigned to **start-protected-client-port**. Note that the maximum supported pool contains 5 entries. Like the protected server port, the protected client port pool should not overlap with the port range defined in the steering ports configuration using the same IP address and the SIP interface. If there is overlap, the NAT table entry for the steering port used in a call will prevent SIP messages from reaching the system's host processor.

   This parameter defaults to 0, which disables the function associated with the parameter. The valid range for values is 1025 to 65535.

6. **encr-alg-list**—Enter the list of encryption algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   This parameter defaults to the following three values: **aes-cbc**, **des-ede3-cbc**, and **null**.

7. **auth-alg-list**—Enter the list of authentication algorithms. You enter more than one value by separating the algorithms by <Spaces> and enclosing all values in quotations marks:

   This parameter defaults to hmac-sha-1-96.

Sample IMS-AKA Configuration

The following formatted extract from **show running-config** ACLI output shows a sample IMS-AKA profile configuration.

```
ims-aka-profile
name    TS33.203
start-protected-client-port    4060
end-protected-client-port    4064
protected-server-port    4070
encr-alg-list    aes-cbc des-ede3-cbc null
auth-alg-list    hmac-sha-1-96 hmac-md5-96
last-modified-by    admin@172.30.11.18
last-modified-date    2013-06-15 14:58:08
```

SIP IMS P-CSCF P-Asserted Identity in Responses

In releases earlier than Release S-C6.1.0, the Oracle USM—operating as a P-CSCF—removes the P-Preferred-Identity header (if present) on receipt of a 1xx or 2xx response. It also inserts a P-Asserted-Identity header with the value received in the P-Preferred-Identity header.

Release S-C6.1.0 changes this behavior. Now the Oracle USM:

- Caches a copy of the P-Called-Party-ID header when it receives one of the following destined for a UE prior to forwarding the request:
  - An initial request for dialog
  - A request for a standalone transaction
  - A request for an unknown method that does not related to an existing dialog

The SIP interface receiving the request should have the SIP IMS feature enabled.

- Removes the P-Preferred-Identity header (if present) and inserts a P-Asserted-Identity header with the value saved from the P-Called-Party-ID header on receipt of a 1xx or 2xx response.

Important Notes

Note the following:

- The endpoint to which the response is being sent must be a trusted endpoint. The option **disable-ppi-to-pai** should not be configured in the global SIP configuration's **options** list.
- If the P-Preferred-Identity header is present in the response, the Oracle USM will delete the header.
- If the P-Asserted-Identity header is present in the response, the Oracle USM will overwrite that -Asserted-Identity.

SIP IMS P-CSCF P-Asserted Identity in Responses Configuration

This behavior is enabled automatically. You do not need to perform any configuration steps.

E-CSCF Support

An Emergency Call Session Control Function (E-CSCF) is an IMS core element that aids in routing emergency calls to an appropriate destination, such as a PSAP. E-CSCF functionality can be performed by the Oracle USM with appropriate local policy and network management control configuration.

The E-CSCF feature let the Oracle USM internally prioritize and route emergency calls to the corresponding Emergency Service Center, based either on the calling party's request URI, or based on location information retrieved from a CLF (Connectivity Location Function) for wireline/TISPAN networks.

By integrating E-CSCF functionality into the P-CSCF (Oracle USM), networks can satisfy the common local requirement that certain telephony elements be deployed locally, rather than use single, centralized elements. Funcitons like the E-CSCF likely fall into this category.

Service URN Support

To enable E-CSCF functionality, the Oracle USM can parse service URNs for local policy lookup keys, and as desitnation identifiers in network management controls (NMC). Ensure that the match-URN is entered correctly as: urn:service:sos" or "urn:service:sos.type or the Oracle USM will interpret the URN as a hostname. Please see RFC 5031 for more information on compliant URN construction.

E-CSCF Configuration Architecture

There are four elements which comprise and enable E-CSCF support on the Oracle USM :

- CLF Connectivity
- NMC Emergency Call Control

- Local Policy
- Emergency Local Route Table

CLF Connectivity

The Oracle USM must be configured with Diameter-based CLF support. This is accomplished by creating an appropriate external policy server configuration.

When the Oracle USM requests authorization from the CLF server, a Line-Identifier AVP which includes a location string is expected to be returned for the call. The returned location string will be used later for an LRT query.

NMC Emergency Call Control

By configuring a Network Management Control (NMC), the Oracle USM can flag a call for special priority early after it is received and validated by the system. The **destination identifier** must be configured in the NMC with the service URN of an incoming emergency call. Also, the NMC configuration must have its **next hop** parameter left blank. This lets the Oracle USM route the emergency call with local policies.

For example, if **urn:service:sos** is the configured value in the NMC's **destination identifier**, and an INVITE arrives on the Oracle USM with urn:service:sos in the request URI, the call will be flagged for emergency handling. The next step in call processing is for the INVITE to be evaluated by local policy.

Local Policy

Local policies must be configured to match and then route an incoming emergency call. Once a local policy match is made, the Oracle USM looks to the configured policy attributes for where to forward the INVITE. A matching policy attribute's next hop should be configured to point to an emergency LRT that contains specific destinations for emergency calls. In addition, the **elec str lkup** parameter must be set to enabled so the Oracle USM will perform an LRT lookup based on the location string returned in the CLF response.

The **eloc str match** parameter identifies the attribute, whose value in the location string will be used as the lookup key in the emergency LRT. For example, if the returned location string is:

```
loc=xxx;noc=yyyy;line-code=zzzz
```

and the **eloc str match** parameter is set to **noc**, then when the Oracle USM performs a local policy route search, it will search the LRT for yyyy. If the **eloc str match** parameter left empty or if there is no match when **eloc str lkup** is enabled, the entire location string is used as the lookup key.

Emergency LRT

The Oracle USM needs to be configured with an emergency LRT to route emergency calls to their destination.

As stated in the previous section, when searching an emergency LRT, any user defined parameter within a Location String may be used as the key to look up next-hop routing information.

LRT files support <user type = string> which enables the Oracle USM to perform searches on free form attributes that may appear in the returned location-string. The <user type = string>

value for an entry in the emergency LRT should be set to a part or whole value returned in the CLF's location string. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<localRoutes>
    <route>
        <user type="string">1234</user>
        <next type="regex">!^.*$!sip:911@192.168.200.140:5060!</next>
    </route>
    <route>
        <user type="string">loc=xxx;noc=yyyy;line-code=zzzz</user>
        <next type="regex">!^.*$!sip:911@192.168.1.139:5060!</next>
    </route>
</localRoutes>
```

> **Note:**
>
> Given that the Location String is not a well-defined string, care should be taken when defining and configuring the LRT tables.

LRTs must be individually uploaded to both the active and standby systems in an HA node; LRTs are not automatically replicated across nodes.

CLF Response Failure

If there is no location string in a CLF's repsone or the CLF rejects the call, the Oracle USM uses the **default location string** parameter from the ingress SIP interface to populate the PANI header. The emergency call proceeds normally using this location string's information for emergency LRT lookups.

E-CSCF Configuration

This procedure assumes that the Oracle USM is configured to communicate with a CLF. In addition, this procedure assumes an the Oracle USM is configured and loaded with an appropriate LRT for E-CSCF Use.

To configure an NMC for E-CSCF use (baseline parameters are not mentioned):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ```

4. **name**—Enter the name of this network management control rule; this value uniquely identifies the control rule. There is no default for this parameter.

5. **state**—Enable or disable this network management control rule. The default value is **enabled**. The valid values are:

   - enabled | disabled

6. **type—**Set this parameter to **priority** so that the Oracle USM will flag incoming calls with a matching destination identifier as a priority calls.
7. **treatment**—Set this parameter to **divert**.
8. **next-hop**—Leave this parameter blank so that the call's processing will go directly to local policy.
9. **destination-identifier**—Enter the service URN that endpoints in your network include in their request URIs to identify themselves as emergency calls.
10. Save your configuration.

    To configure local policy for E-CSCF use (baseline parameters are not mentioned):

11. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

12. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```

13. Type **local-policy** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

    ```
    ORACLE(session-router)# local-policy
    ORACLE(local-policy)#
    ```

14. **to-address**—Set this parameter to the lookup key for matching emergency calls. You can now use a service URN as lookup key criteria.
15. Save your configuration.

    To configure policy attributes for E-CSCF use (baseline parameters are not mentioned):

16. Type **policy-attributes** and press Enter. If you are adding this feature to a pre-existing local policy configuration, you will need to select and edit your local policy.

    ```
    ORACLE(local-policy))# policy-attributes
    ORACLE(policy-attributes)#
    ```

17. **next-hop**—Set this parameter to **lrt: name-of-elrt-file.gz** for this policy attribute to lookup routes in the named lrt file.
18. **eloc-str-lkup**—Set this parameter to **enabled** for the Oracle USM to parse the emergency location string, as received in a CLF Line Identifyier AVP, for emergency LRT lookup.
19. **eloc-str-match**—Set this parameter to the attribute name found in the location string whose value will be used as a lookup key in the LRT named in the next-hop parameter. Common values include "loc" or noc.
20. Save and activate your configuration.

Maintenance and Troubleshooting

The **show lrt route-entry** command displays two entries, if the username 1234 has a "string" type and "E164" type entries.

```
ACMEPACKET# show lrt route-entry emergency_lrt 1234
UserName <1234>
 User Type= E164
 NextHop= !^.*$!sip:911@192.168.200.139:5060!
 NextHop Type= regexp
```

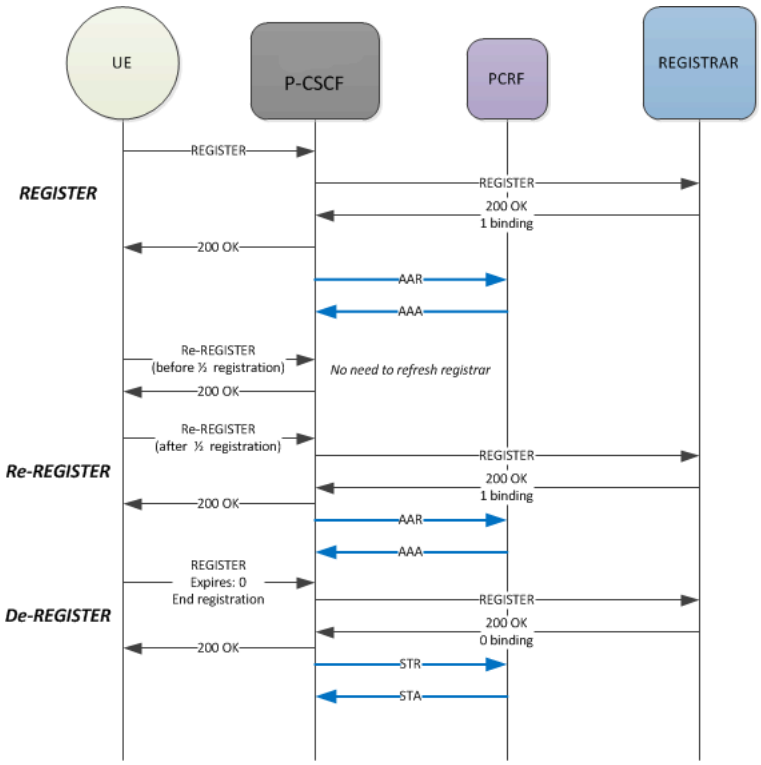
```
UserName <1234>
 User Type= string
 NextHop= !^.*$!sip:911@192.168.200.140:5060!
 NextHop Type= regexp
```

2774 - Provisioning of SIP Signaling Flow Information

This feature supports of Provisioning of SIP Signalling Flow Information as described in 3GPP TS 29.213 section B1b [1], and the procedures specified in TS 29.214 section 4.4.5a.

The feature applies to the scenario when the Oracle USM (A-SBC / P-CSCF) uses its external policy server functionality and connects to a PCRF via Rx interface in bandwidth-management mode.

This feature deals with the information that the Oracle USM includes in an AAR message it sends to the PCRF when an endpoint registers, re-registers, and de-registers. The following diagram shows the typical scenario.



Initial Registration

When an endpoint registers, the Oracle USM creates and sends an AAR message to a PCRF which includes the following information:

<AA-Request>::= < Diameter Header: 265, REQ, PXY >

AVP	AVP Contents
Session-ID	OriginHost;0;0;<Key to Registration Cache> Key is usually the endpoint's AoR.
Auth-Application-ID	Application-ID of Rx (16777236)
Origin-Host	<ext-policy-server_name>.<ext-policy-realm>.<domain-name-suffix>
Origin-Realm	<ext-policy-realm>.<domain-name-suffix>
Destination-Realm	<IP Address of endpoint>@<destination realm of this AAR>
Framed-IP-Address AVP - v4 address Framed-IPv6-Prefix AVP - v6 address	Layer 3 Endpoint-IP-address v4 - Framed-IP-Addressor v6 - Framed-IPv6-Prefix
Media-Component	Grouped AVP description follows

Media-Component-Description AVP ::= < AVP Header: 517 >

AVP	AVP Contents
Media-Component-Number	0
Media-Sub-Component	Grouped AVP description follows

Media-Sub-Component ::= < AVP Header: 519 >

AVP	AVP Contents
Flow-Number	1
Flow-Description	Permit in <ip> from <Endpoint IP:Port> to <USM Sip Interface IP:Port> Permit out <ip> from <USM SIP Interface IP:Port> to <Endpoint IP:Port> Where <ip> is (UDP: 17, TCP: 6) if wildcard-trans-protocol = disabled.
Flow-Status	Set to: ENABLED (2)
Flow-Usage	Set to: AF_SIGNALLING (2)
AF-Signalling-Protocol	Set to: SIP (1)

Register Refresh

When a registration Refresh is received before the half time of the registration expiry, the registration cache is not updated and the Oracle USM responds with 200OK to the UE. This is standard registration cache functionality. No additional AAR is sent to the PCRF.

If a registration Refresh is received after the half time of the registration expiry, or if any registration information changes, the Oracle USM sends an AAR to the PCRF after it receives and forwards a 200 OK response from registrar. The AAR includes the same session-id as the initial AAR that was to PCRF. This lets PCRF correlate the AAR with the earlier AAR that it received.

De-Registration

When an contact de-registers with expires=0, it means that the endpoint is removing all of its contacts from registration. When this occurs, the Oracle USM sends an STR message to the PCRF to terminate the session. If the de-registration message does not reflect a complete de-

registration, the Oracle USM does not send the STR message to the PCRF. The STR message includes the following information:

<ST-Request> ::= < Diameter Header: 275, REQ, PXY >

AVP	AVP Contents
Session-ID	OriginHost;0;0;<Key to Registration Cache> Key is usually the endpoint's AoR.
Auth-Application-ID	Application-ID of Rx (16777236)
Origin-Host	<ext-policy-server_name>.<ext-policy-realm>.<domain-name-suffix>
Origin-Realm	<ext-policy-realm>.<domain-name-suffix>
Destination-Realm	<IP Address of endpoint>@<destination realm of this AAR>
Destination-Host	
Termination-Cause	DIAMETER LOGOUT (1) - User initiated a disconnect

Failure Response to Re-Register

Upon reception of a failure response from the REGISTRAR for a subsequent Registration refresh from endpoint, the Oracle USM performs de-registration actions, i.e. an STR message to the PCRF.

Provisioning SIP Signaling Flows Configuration

To enable correct provisioning of signaling flows upon registration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **ext-policy-server** and press Enter.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the external policy server that you want to edit.

4. **provision-signaling-flow**—Set this to enabled for the Oracle USM to send AAR messages to a PCRF on endpoint registrations.
5. Save and activate your work.

Troubleshooting

Upon receipt of AAA response from PCRF for the AAR sent, the Oracle USM logs the status of the AAA received. Successful AAAs are noted by session-ID in EBMD logs. The log will look like:

```
Provision of SIP Signaling Flow (<session-ID>) via Diameter Rx on realm <realm-
id> successful.
```

Failed AAA response are noted at NOTICE level EBMD logs citing the failure response to AAR request with the session-ID sent. An a MINOR non-health-affecting alarm is also raised with the following text:

```
Provision of SIP Signaling Flow (<session-ID>) via Diameter Rx on realm <realm-
id> failed.
```

show ext-band-mgr

The show ext-band-mgr command has been augmented to include the Register and DeRegister counts. For example:

```
ORACLE# show ext-band-mgr
13:55:49-166
EBM Status                  -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          7       4       1
Server Trans           0       0       0          0       0       0
Sockets                1       1       0          1       1       1
Connections            0       0       0          1       1       1
                           ---- Lifetime ----
                     Recent      Total  PerMax
Reserve                   0          0       0
Modify                    0          0       0
Commit                    0          4       2
Remove                    0          2       1
Register                  1          1       1
DeRegister                1          1       1
EBM Requests              0          6       3
EBM Installs              0          6       3
EBM Req. Errors           0          0       0
EBM Rejects               0          0       0
EBM Expires               0          0       0
EBMD Errors               0          0       0
```

RTP and RTCP Bandwidth Calculation and Reporting

The Oracle USM supports changing bandwidth requirements in an ad-hoc mult-party conference by tracking reduced bandwidth needs as parties are placed on hold during the initiation of a multi-party call. The combination of the 5 AVPs are considered by network elements for this functionality.

This section is applicable to the Oracle USM's Rx implementation when acting as a P-CSCF and connecting with a PCRF. The 5 AVPs considered in this section are created and sent in AAR messages.

Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs

These AVPs reflect RTP bandwidth requirements for a call. The default behavior (i.e., no options are configured) that dictates how these AVPs are created follows.

The average rate limit scenario is when no b=AS: parameter in the SDP and the media-profile > **average-rate-limit** parameter is configured to a value greater than 0. The Oracle USM inserts the media-profile > **average-rate-limit** parameter ×8 into both Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs.

The SDP bandwidth scenario is when there is a b=AS: parameter in the SDP and the media-profile configuration element has the following configurations:

- average-rate-limit = 0
- sdp-rate-limit-headroom > 0
- sdp-bandwidth = ENABLED

When these conditions are met, the Max-Requested-Bandwidth-UL & Max-Requested-Bandwidth-DL AVPs are populated with the value in the b=AS: parameter + **sdp-rate-limit-headroom** ACLI parameter.

Optional AVP Creation

When the **get-bw-from-sdp** option is configured in the sip-config, the following occurs:

When SDP contains the

`b=AS:`

parameter with valid value, the Oracle USM multiplies that value ×1000 and inserts the result in the Max-Requested-Bandwidth-UL and Max-Requested-Bandwidth-DL AVPs in an AAR message.

If there is no b=AS: line in the SDP, the value is taken from the media-profile > **average-rate-limit** parameter multiplied ×8 to get a bps value then inserted into the Max-Requested-Bandwidth-UL and Max-Requested-Bandwidth-DL AVPs in an AAR message.

The Oracle USM uses the b=AS: line (or chosen codec via the media-profile) from the final answer SDP for the session as the value for creation of these two AVPs.

RR-Bandwidth & RS-Bandwidth AVPs

These AVPs reflect RTCP bandwidth requirements for a call. When SDP contains:

```
b=RR:
b=RS:
```

parameters and values, the Oracle USM inserts the values (after the :) into the respective RR-Bandwidth (521) and RS-Bandwith (522) AVPs. When these parameters are not present in the SDP, the RR-Bandwidth and RS-Bandwidth AVPs are not created in AAR messages sent to the PCRF. The sip-config > get-bw-from-sdp option must be configured to enable creation of these AVPs.

Flow-status AVP

The Flow-status AVP (511) is based on SDP direction. Tests for SDP are performed in the following order:

SDP State	Flow-Status AVP (511) Set to
port in the SDP m-line is 0	REMOVED (4)
Transport in m-line is "TCP" or " TCP/MSRP"	ENABLED (2)
a=recvonly and <SDP direction> = UE originated	ENABLED-DOWNLINK (1)
a=recvonly and <SDP direction> = UE terminated	ENABLED-UPLINK (0)
a=sendonly and <SDP direction> = UE originated	ENABLED-UPLINK (0)

SDP State	Flow-Status AVP (511) Set to
a=sendonly and <SDP direction> = UE terminated	ENABLED-DOWNLINK (1)
a=inactive	DISABLED (3)
a=sendrecv or no direction attribute	ENABLED (2)

UE originated - Call originator creates the SDP being considered.

UE terminated - Call terminator creates the SDP being considered.

Flow-status AVP is always created according and requires no configuration.

2629 - IR.92 Compliance via SIP 380 Response

This feature furthers the Oracle USM's compliance with GSMA's Voice over LTE specification (IR.92) to redirect VoLTE originated emergency calls to a circuit switched network.

When the Oracle USM receives an emergency call, which it can not complete, it returns a 380 (Alternative Service) response to the sender. Some examples of when the Oracle USM can not forward such an emergency call are:

- The Next-hop does not exist to route the call via NMC means
- The NSEP calls are rejected due to the Oracle USM hitting a load limit
- The NSEP calls are rejected due to the target session agent exceeding constraints
- The NMC treatment for the call is set to Reject

380 Response Format

The Oracle USM's 380 SIP response to the sender includes:

- Content-Type header field set to application/3gpp-ims+xml
- P-Asserted-Identity header field set to the value of the SIP URI of the last entry on the Path header field value received during registration; It is the value of the SIP URI of the P-CSCF.
- 3GPP IM CN subsystem XML body containing an <ims-3gpp> element with the "version" attribute will be set to "1" and with an <alternative-service> child element set to "alternative service"
  - a <type> child element set to emergency
  - a <reason> child element set to the value of configuration element send-380-response
  - an <action> child element set to emergency-registration

380 Response Example

```
SIP/2.0 380 Alternative Service
Via: SIP/2.0/UDP 192.168.15.2:5060;branch=z9hG4bK-23615-1-0
From: sipp <sip:911@192.168.15.2:5060>;tag=1
To: sut <sip:911@192.168.101.11:5060>
Call-ID: 1-23615@192.168.15.2
CSeq: 1 INVITE
Content-Type: application/3gpp-ims+xml
Content-Length: 209
```

```
P-Asserted-Identity: sip:911-44e2etbufgibf@172.16.101.11:5060
Reason: Q.850; cause=63
<?xml version='1.0' encoding="UTF-8"?>
<ims-3gpp version="1.0">
<alternative-service>
<type>emergency</type>
<action>emergency-registration</action>
<reason>sample reason</reason>
</alternative-service>
</ims-3gpp>
```

IR.92 Compliance Configuration

To configure the 380 SIP response for IR.92 compliance:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.
5. **send-380-response**—Enter a reason phrase enclosed in quotes to place in the reason tag of the <ims-3gpp> element in a 380 response for a failed-to-route emergency call.
6. Type **done** and continue.

IR.94 Support

The Oracle USM supports IR.94 (IMS Profile for Conversational Video Service) for use in an IMS and LTE environment.

The Oracle USM supports both video and audio and can keep a call up by switching to audio mode when the video session is dropped. The Oracle USM receives a Specific-Action AVP in the RAR from the PCRF indicating the loss of video bearer. The Oracle USM then replies with a RAA, and finally sends a STR failing the call.

This behavior contradicts TS 29.214 that indicates the P-CSCF should wait for an ASR from the PCRF (indicating the bearer has been removed), and should not fail the call until all bearers have been removed.

The following behaviors enable IR.94 Support:

- The Oracle USM supports the IR.94 re-INVITE procedures for adding or removing video mid-call.
- The Oracle USM recognizes that a UE is capable of handling video calls when in the REGISTER message, the sip.video media feature is present in the Contact: header.

- The Oracle USM supports Audio-Video Profile with Feedback (AVPF) and SDP Capability Negotiation (RFC 5939) as documented in TS 26.114 section 6.2.1a.
- When the Oracle USM supports sessions (200 OK received after an INVITE) with a single media stream throughout the session duration, it may receive any of the following notifications in the Specific-Action AVP of the RAR from the PCRF:
  - INDICATION_OF_LOSS_OF_BEARER (2)
  - INDICATION_OF_RELEASE_OF_BEARER (4)
  - INDICATION_OF_OUT_OF_CREDIT (7)
  - INDICATION_OF_FAILED_RESOURCES_ALLOCATION (9)

  The Oracle USM then:

  1. Replies with an RAA acknowledging the RAR
  2. Sends a BYE to both the UE and the Core, then
  3. Sends an STR to the PCRF indicating the session has been terminated.
- When the Oracle USM supports sessions (200 OK received after an INVITE) with multiple simultaneous media streams sometime during the session, it may receive any of the following notifications in the Specific-Action AVP of the RAR from the PCRF:
  - INDICATION_OF_LOSS_OF_BEARER (2)
  - INDICATION_OF_RELEASE_OF_BEARER (4)
  - INDICATION_OF_OUT_OF_CREDIT (7)
  - INDICATION_OF_FAILED_RESOURCES_ALLOCATION (9)

  The Oracle USM then:

  1. Sends an RAA acknowledging the RAR, and no further action is required When an eSR-VCC handover occurs, the ATCF (Oracle USM) sends a INVITE/UPDATE without video (voice-only SDP) toward the target UE.

IR.94 Loss Of Voice Bearer

The Oracle USM provides compliance with an IMS Profile for Conversational Video Service (IR.94) requirement that specifies the termination of a multi-media session (voice and video) if the voice bearer is lost.

Version S-CZ7.2.0 adds a new parameter (options **terminate-on-voice-bearer-release**) that addresses an unlikely, but possible, corner case in which the voice bearer is lost but the video bearer remains in service. Prior to Version S-CZ7.2.0, the Oracle USM would retain the call as a video-only session. This behavior is not compliant with IR.94, which specifies that the video-only session should not be allowed to continue.

The option **terminate-on-voice-bearer-release** enables compliance with the IR.94 standard. Compliance is ensured by basing the decision to terminate or continue a multi-media session on the state of the voice bearer exclusively; the state of any other media bearer (video, for example) plays no role in the decision process. Consequently, if the voice bearer fails, the call terminates; failure of the video bearer, or other media stream, is not pertinent to call termination or continuance.

IR.94 Loss Of Voice Bearer Configuration

Use the following procedure to enable IR.94 compliance.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. Use options terminate-on-voice-bearer-release to enable I.94 compliance.

```
ACMEPACKET(ext-policy-server)# options +terminate-on-voice-bearer-release+
    ACMEPACKET(ext-policy-server)#
```

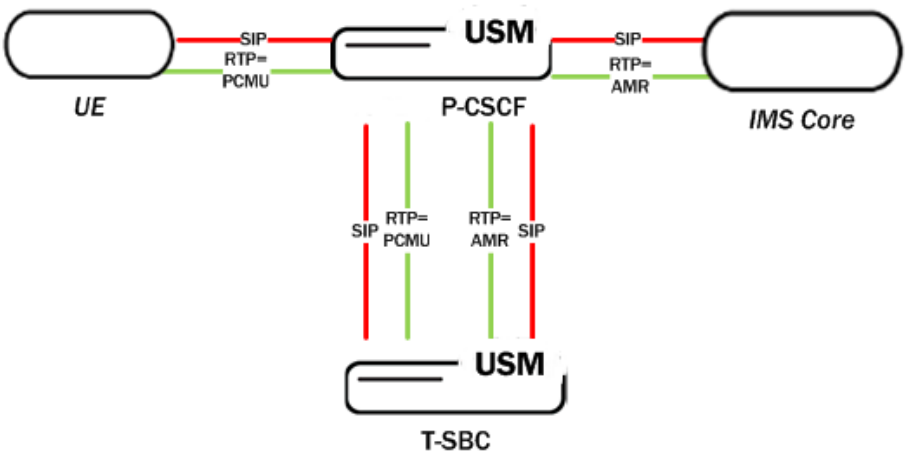
4. Type **done** to save your configuration.

Pooled Transcoding

The term pooled transcoding refers to a deployment model for IMS environments involving two or more Oracle USMs. The first is an Oracle USM acting as the P-CSCF, and the others are one or more Oracle USMs deployed with transcoding capabilities (referred to as T-SBCs). The T-SBC provides transcoding resources—a pool—that the Oracle USM can invoke on-demand.

In the pooled transcoding model, the Oracle USM sits between realms or between user endpoints that require transcoding between their preferred codecs to communicate. This deployment model conserves resources on both the Oracle USM and the T-SBC. While the Oracle USM serves as the access function with encryption support, the T-SBC supports transcoding in a tunneling gateway (TG) configuration to meet high-density transcoding requirements.

The following diagram shows an Oracle USM positioned between an access UE and the IMS core. The Oracle USM compares SDP offers and answers from the elements it sits between, and uses the results to determine whether or not a given session requires transcoding. If transcoding is required, the Oracle USM invokes T-SBC's services. Acting as a B2BUA, T-SBC uses information from the P-CSCF's SIP messaging to transcode between the applicable codecs and then to route SIP signaling back to the P-CSCF on the egress.



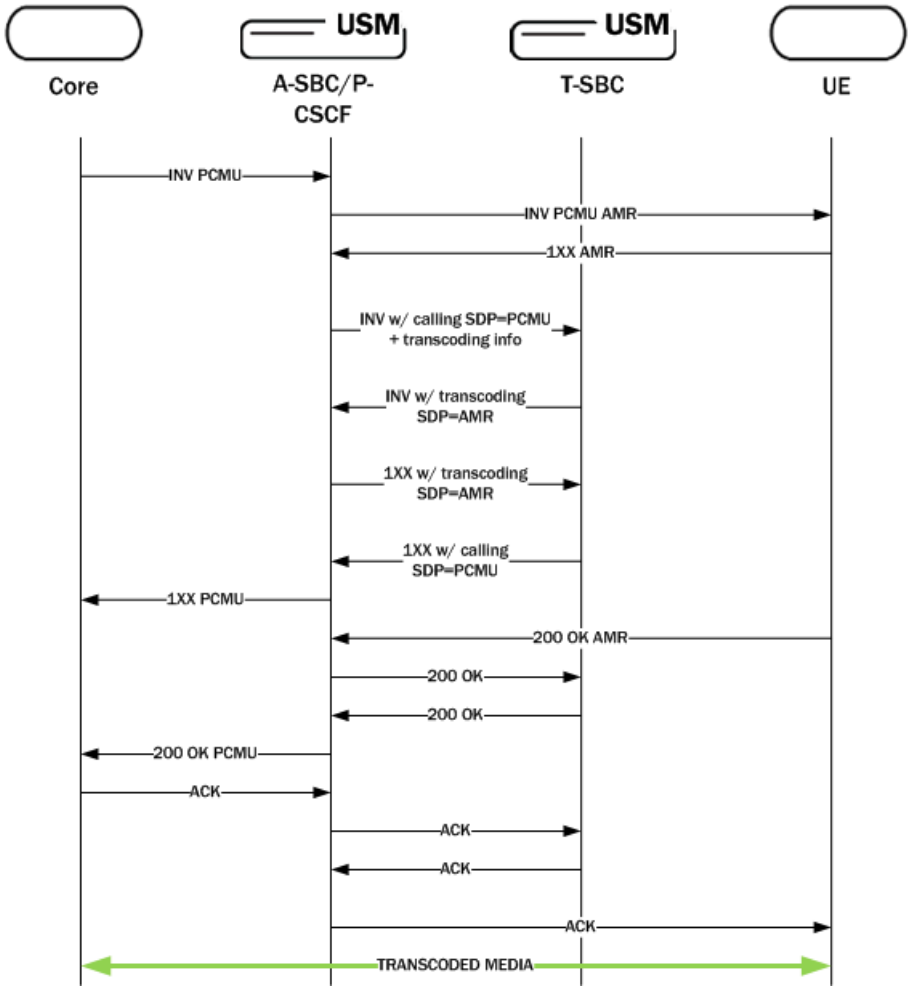
When a call comes in, the system creates two ports for the caller and callee realms respectively. When the call needs pooled transcoding, Oracle USM creates four ports on the pooled transcoded realm and two additional ports on the caller realm. All the ports are removed at the end of the call.

For example, for a call between the realms net172 (Offer) and net145 (Answer) :

Before pooled transcoding	During pooled transcoding	After pooled transcoding
Net172 -> 2 Ports	Net172 -> 4 Ports	Net172 -> 2 Ports
Net145 -> 2 Ports	Net145 -> 2 Ports	Net145 -> 2 Ports
Net192 (Transcoding Realm) -> 0 Ports	Net192 (Transcoding Realm) -> 4 Ports	Net192 (Transcoding Realm) -> 0 Ports

In the previous example, the USM uses four ports on Net192 are for communicating with the transcoding USM. The USM uses the two newly created ports on Net172 for RTP communication, while the original two ports are not used.

The following diagram shows what a call flow between entities in such a deployment model looks like:



Supported Codecs

The is a list of transcodable codecs Oracle USM supports. The pooled transcoding deployment model also supports transcoding for these codecs.

- PCMU
- CMA
- G729
- G729A
- iLBC
- telephone-event
- G726
- G726-16
- G726-24
- G726-32
- G726-40
- G722
- G723
- GSM
- AMR
- AMR-WB

Implementation Details

From the Oracle USM (P-CSCF) perspective, the T-SBC is represented as a transcoding agent whose services the P-CSCF can invoke when offer-answer exchanges reveal transcoding is required. Once that determination is made, the P-CSCF initiates communication with the T-SBC. The P-CSCF uses its public SIP interface and a corresponding realm reserved for communication with a T-SBC, which is configured as a transcoding agent on the P-CSCF. Multiple transcoding agents can be configured, and can be IP addresses, session agents, or session agent groups (SAGs).

If there is more than one transcoding agent for the P-CSCF to choose from, the P-CSCF bases its selection on the order in which the transcoding agents were entered in the list. When conditions call for it, the P-CSCF will try each transcoding agent listed one by one until it:

- Receives a 2xx response from a transcoding agent,
- The list is exhausted, or
- The original transaction times out.

If the transcoding agents are session agents with hostnames, the P-CSCF uses DNS to resolve the hostnames and then tries the hosts in order. In the case when transcoding agents are SAGs, the P-CSCF selects the session agent according to the selection strategy configured for the SAG. The P-CSCF will recurse through all members of the SAG when SAG recursing is enabled. Whenever session agents and SAGs do not have ports or transport protocols specified, the defaults are 5060 and UDP respectively.

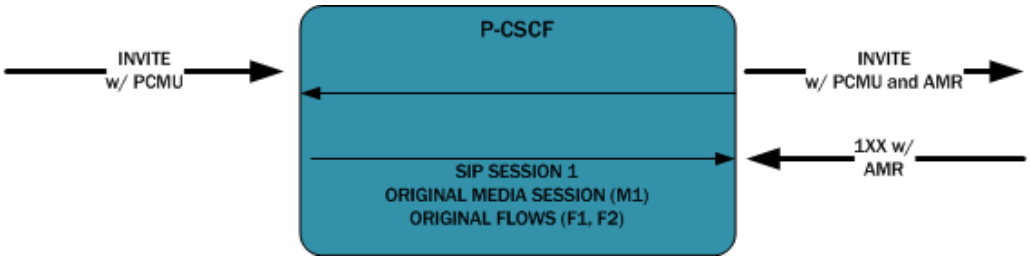
Once it identifies a transcoding agent, the P-CSCF sends an INVITE to which the transcoding agent—the T-SBC—responds with a 2xx message. Configuring the P-CSCFs as a session agent and disabling dialog transparency (in global SIP configuration) on the T-SBC allows it to accept SIP messages from the P-CSCF. Then the T-SBC acts as a B2BUA, using the information received in the P-CSCF's INVITEs to invoke transcoding and to route SIP signaling back to the P-CSCF on the egress.

Application Scenarios

This section discusses two application scenarios for pooled transcoding, one for how the P-CSCF handles an INVITE with SDP and one for how it handles an INVITE without SDP.

Scenario 1 INVITE with SDP

When the P-CSCF receives an INVITE with SDP, it creates a SIP session and an associated media session that has two flows for audio. The P-CSCF applies the appropriate codec policy that has the **add-on-egress** configured, so the egress INVITE has this codec present.



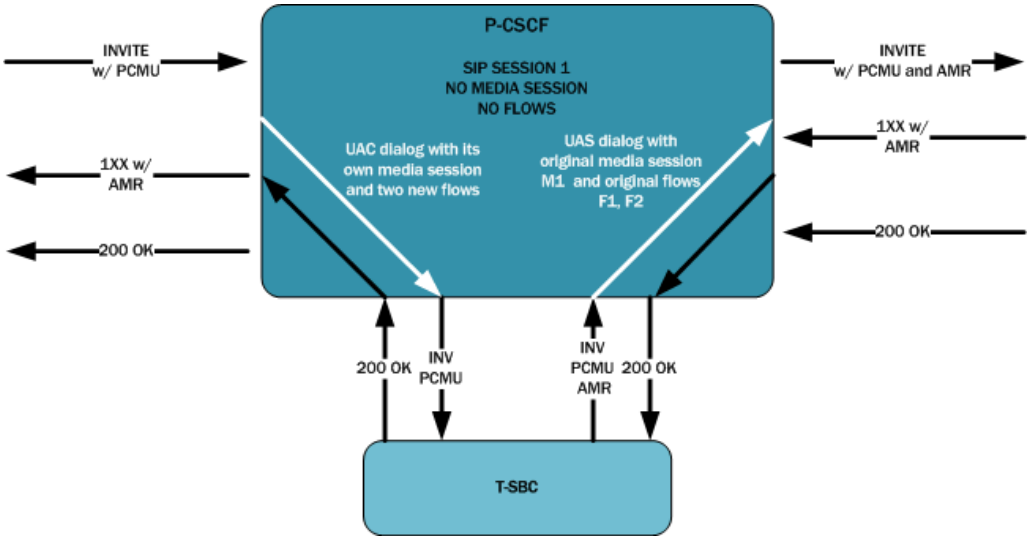
The P-CSCF receives an answer to this INVITE, and if it contains the added codec, the P-CSCF invokes the T-SBC's transcoding services using an INVITE with the same SDP as the INVITE received on ingress. The communication between the P-CSCF and the T-SBC is a stand-alone dialog, and a new media session is associated with it.

This is an example of what the INVITE the P-CSCF sends to the T-SBC looks like:

```
INVITE sip:192.168.101.78:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.18:5060;branch=z9hG4bK10dspa3058b1io4rk721
Max-Forwards: 70
Call-ID: 9fc71d79b43b4b95de41acefd71a8bc4@192.168.101.78
To: sip:192.168.101.78:5060
Contact: sip:172.16.101.18
From: sip:172.16.101.18;tag=bcf80756721880b7996ccb488b6a1b2f
CSeq: 1 INVITE
Target-Dialog: 1-16881@172.16.18.5;local-tag=1;remote-
tag=16880SIPpTag011;realm=net192
Acme-Codec-Policy: ;ingress;name="in-2833";allow-codecs="* ";add-codecs-on-
egress="";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Codec-Policy: ;egress;name="out-2833";allow-codecs="* ";add-codecs-on-
egress="PCMA ";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Content-Type: application/sdp
Content-Length: 196^M
P-Visited-Network-ID: open-ims.test
Route: <sip:192.168.101.18:5060;lr;transport=UDP>
v=0
o=user1 53655765 2353687637 IN IP4 192.168.101.18
s=-
c=IN IP4 192.168.101.18
```

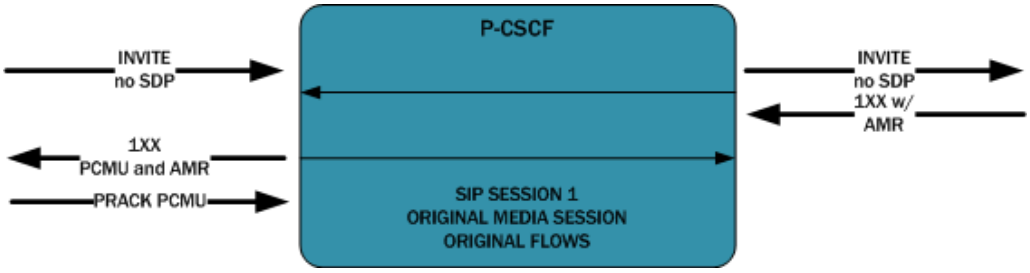
```
t=0 0
m=audio 20002 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
```

Note that the Target Dialog and Acme-Codec-Policy headers communicate operational parameters for pooled transcoding. Using this information, the T-SBC applies two codec policies and returns the INVITE to the P-CSCF. The P-CSCF moves the media session to itself, but the SIP session does not have a media session at this point.



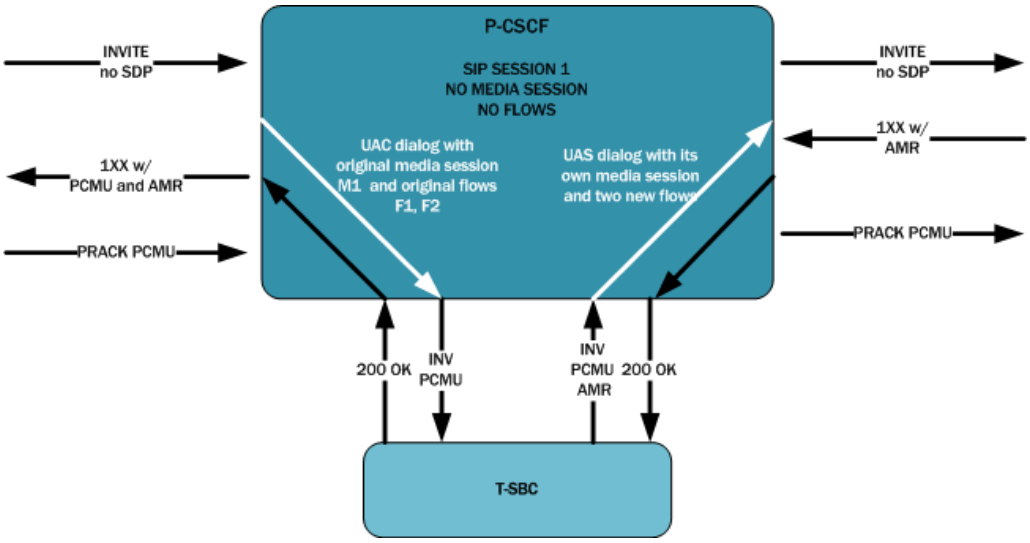
Scenario 2 INVITE without SDP

For an offerless call flow, the media session is created when the offer comes in a reliable provisional or final response. The P-CSCF applies the codec policy and sends the egress offer to the calling UE.



When it receives an answer in either a PRACK or ACK message, the P-CSCF compares the offer and answer. An answer containing the codec added in the egress offer causes the P-CSCF to invoke the T-SBC and create a standalone UAC dialog, just as in Scenario 1.

Since the P-CSCF advertised its media address in the egress offer to the calling UE, the original media session is used by the UAC dialog.



Re-INVITES and Updates with SDP

Once a specific session on the P-CSCF invokes a T-SBC resources, it continues to use the same T-SBC for the session's entire duration. Anytime it receives a SIP message containing modified SDP, the P-CSCF communicates the modification to the T-SBC.

RFC 2833 Considerations

For legacy RFC 2833 interworking to function properly, the P-CSCF communicates the RFC 2833 configuration to the T-SBC in the UAC dialog.

The following is a sample of what a P-CSCF's INVITE with RFC 2833 information sent to the T-SBC looks like:

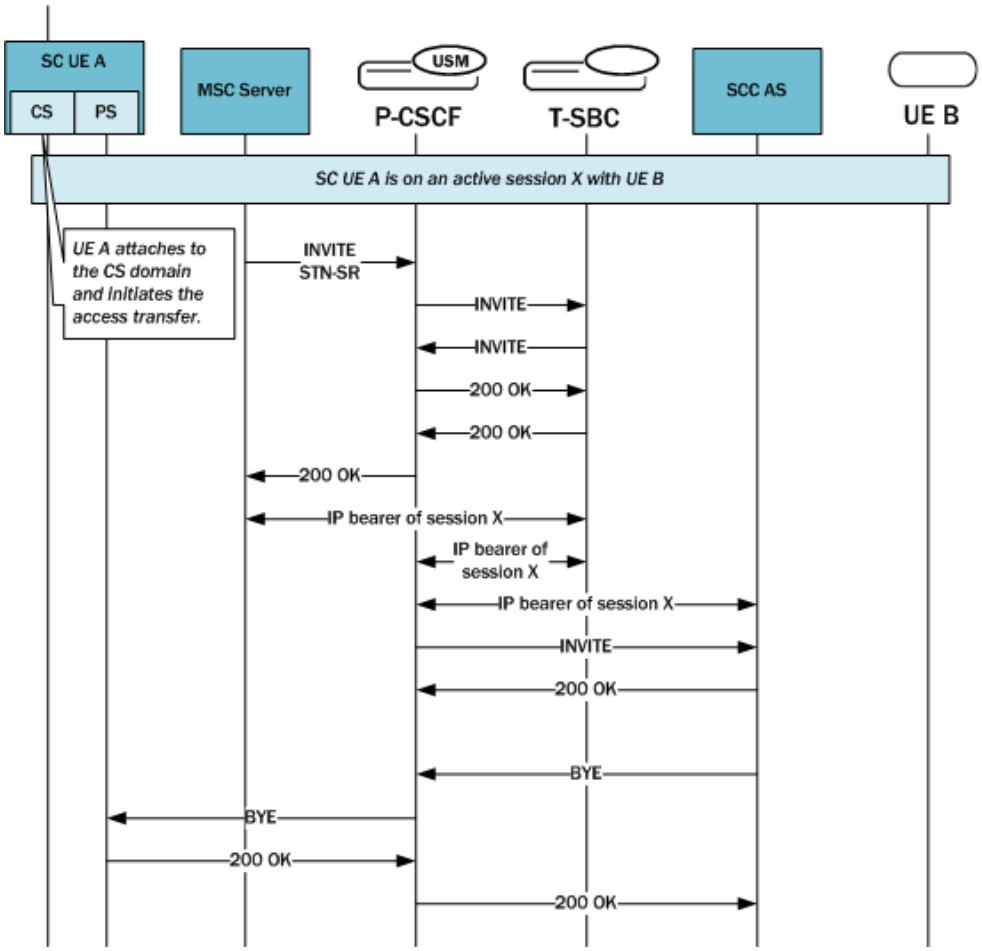
```
INVITE sip:192.168.101.78:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.18:5060;branch=z9hG4bK10dspa3058b1io4rk721
Max-Forwards: 70
Call-ID: 9fc71d79b43b4b95de41acefd71a8bc4@192.168.101.78
To: sip:192.168.101.78:5060
Contact: sip:172.16.101.18
From: sip:172.16.101.18;tag=bcf80756721880b7996ccb488b6a1b2f
CSeq: 1 INVITE
Target-Dialog: 1-16881@172.16.18.5;local-tag=1;remote-
tag=16880SIPpTag011;realm=net192
Acme-Codec-Policy: ;ingress;name="in-2833";allow-codecs="* ";add-codecs-on-
egress="";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Codec-Policy: ;egress;name="out-2833";allow-codecs="* ";add-codecs-on-
egress="PCMA ";force-ptime="disabled";packetization-time="20";dtmf-in-
audio="disabled";order-codecs=""
Acme-Rfc2833: ;ingress;Digit-
Mode=0;PtTo=101;PtFrom=0;TransNon2833=disabled;TransNonInband=disabled
Acme-Rfc2833: ;egress;Digit-Mode=1;PtTo=101;PtFrom=0
Content-Type: application/sdp
Content-Length: 196
P-Visited-Network-ID: open-ims.test
Route: <sip:192.168.101.18:5060;lr;transport=UDP>
v=0
o=user1 53655765 2353687637 IN IP4 192.168.101.18
```

```
s=-
c=IN IP4 192.168.101.18
t=0 0
m=audio 20002 RTP/AVP 0 101
a=rtpmap:0 PCMU/8000
```

eSRVCC Support

For enhanced Signal Radio Voice Call Continuity (eSRVCC), the pooled transcoding deployment model supports instances when:

- Original call is not transcoded, but the handover call is
- Original call is transcoded, but the handover call is not
- Original call is transcoded, and so is handover call



Configuration Requirements and Verification

Pooled transcoding requires specific configuration of the Oracle USM and the T-SBC. Note that the Oracle USM does not enforce these requirements, so set up your pooled transcoding configuration with care. However, verifying your configuration can help to identify issues.

A-SBC P-CSCF Requirements

Your A-SBC/P-CSCF configuration must include the following:

- A public SIP interface the A-SBC/P-CSCF can use for communication with the T-SBC.
- A transcoding realm, which is a separate realm for the public SIP interface and is only used for communication with the T-SBC.
- Appropriate codec policies, including ones set up to add SDP on the egress leg of the session.
- A global SIP configuration, with **transcoding-realm** set to the name where valid transcoding agents reside, and a **transcoding-agents** list configured with one or any combination of: a DNS hostname for a single session agent that can resolve to one or more IP addresses, an IP v4 or v6 address with or without the port specified (if not specified, the default port used is 5060); the name of a session agent group.

T-SBC Requirements

For information about how to configure transcoding functionality on the T-SBC, refer to the . That guide contains information of the system's transcoding capabilities, defining ingress and egress transcoding policies, defining applicable media policies.

Your configuration of the T-SBC must include the following:

- A global SIP configuration, with **dialog-transparency** set to **disabled**—Dialog transparency must be disabled on the T-SBC so the T-SBC's INVITE will have a different callID and From tag.
- A public realm with **codec-manip-in-realm** set to **enabled**, because the same realm is used for transcoding. In addition, the **mm-in-realm** parameter must be enabled.

Configuration Verification

You can verify your configuration using the ACLI **verify-config** command. When verifying your configuration on the A-SBC, errors messages will appear if:

- The **transcoding-realm** value configured is not a valid realm.
- Either one of the **transcoding-realm** or **transcoding-agents** parameters is not configured.
- One or more session agent names defined in the **transcoding-agents** list is not a valid session agent.
- If the IP address version for a agent in the **transcoding-agents** list is not supported in the transcoding realm you identify; e.g., if you list an IPv6 agent in the list and the transcoding realm does not support IPv6, the system will return an error when you verify your configuration.
- If the transcoding agent is a hostname and not a valid session agent.

ACLI Configuration

This section shows you how to configure a transcoding realm and transcoding agents on your A-SBC, acting as the P-CSCF in a pooled transcoding deployment model. You set these as part of the global SIP configuration.

To configure a transcoding realm and transcoding agents for the P-CSCF:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding this support to a pre-existing SIP configuration, you have to select the configuration before you make changes.

4. **transcoding-realm**—Enter the name of a configured realm designated as the separate realm for the public SIP interface, to be used only for communication with the T-SBC in pooled transcoding deployments.

   ```
   ORACLE(sip-config)# transcoding-realm net157
   ORACLE(sip-config)#
   ```

5. **transcoding-agents**—Enter the IP address, IP address and port combination, session agent hostname, or SAG name in this list if you want them to be used as transcoding agents. You can make multiple entries in any combination of these values. For example, you might list an IPv6 address and port, a session agent, and a SAG.

   - To make multiple entries in the list using in one command line, enclose the entire list of value in parentheses (( )), separating each with a Space as in the example below.
   - To add a transcoding agent to an existing list, put a plus sign before the value you want to add, e.g. +154.124.2.8.
   - To remove a transcoding agent from an existing list, put a minus sign before the value you want to remove, e.g. -154.124.2.8.

     ```
     ORACLE(sip-config)# transcoding-agent (sag:sag1 192.168.4.7.3
     192.168.2.6:5061)
     ORACLE(sip-config)#
     ```

6. Save and activate your configuration.

Monitoring P-CSCF—T-SBC Dialogs

You can monitor pooled transcoding communications between the P-CSCF and the T-SBC using the **show sipd pooled-transcoding** ACLI command. The display shows you information for the client and server User Agents on the P-CSCF.

- The UAC Dialogs section shows the number of UAC dialogs the P-CSCF initiated with the T-SBC; the count is cumulative and not for any specific session.
- The UAS Dialogs section shows the number of UAS dialogs the P-CSCF created upon receipt of an INVITE from the T-SBC; the count is cumulative and not for any specific session.
- The XCodeSessions counts the number of transcoded sessions on the P-CSCF.

```
ACMEPACKET# show sipd pooled-transcoding
18:11:28-125
SIP Pooled Transcoding Status
                    -- Period --                  -------- Lifetime --------
                Active    High   Total      Total  PerMax    High
UAC Dialogs          1       1       1          1       1       1
  Early              0       0       0          0       0       0
  Confirmed          1       1       1          1       1       1
  Terminated         0       0       0          0       0       0
UAS Dialogs          1       1       1          1       1       1
  Early              0       0       0          0       0       0
  Confirmed          1       1       1          1       1       1
  Terminated         0       0       0          0       0       0
XCodeSessions        1       1       1          1       1       1
```

Per-Method Statistics

If you set **extra-method-stats** to **disabled** in your global SIP configuration, you can display the P-CSCF's per-method statistics for the public transcoding-realm.

```
ACMEPACKET# show sipd realms net157 invite
INVITE (18:14:59-36)
                        --------- Server --------   --------- Client --------
Message/Event           Recent      Total  PerMax   Recent      Total  PerMax
                        ------  ---------  ------   ------  ---------  ------
INVITE Requests              0          1       1        0          1       1
Retransmissions              0          0       0        0          0       0
100 Trying                   0          1       1        0          1       1
200 OK                       0          1       1        0          1       1
Response Retrans             0          0       0        0          0       0
Transaction Timeouts         -          -       -        0          0       0
Locally Throttled            -          -       -        0          0       0

Avg Latency=0.000 for 0
Max Latency=0.000
BurstRate Incoming=1 Outgoing=0

ACMEPACKET# show sipd realms net157 bye
BYE (18:15:13-50)
                        --------- Server --------   --------- Client --------
Message/Event           Recent      Total  PerMax   Recent      Total  PerMax
                        ------  ---------  ------   ------  ---------  ------
BYE Requests                 0          1       1        0          1       1
Retransmissions              0          0       0        0          0       0
200 OK                       0          1       1        0          1       1
Transaction Timeouts         -          -       -        0          0       0
Locally Throttled            -          -       -        0          0       0

Avg Latency=0.000 for 0
Max Latency=0.000
BurstRate Incoming=1 Outgoing=0
```

Notes on the DIAMETER Rx Interface

DIAMETER Rx support for external bandwidth management is specialized for pooled transcoding.

For pooled transcoding, the identifier appears in this altered format: <policy server name>;<policy server realm>;<dns suffix>. Because there are two dialogs (one for the server UA and one for the client UA), there are two separate media sessions. This situation requires the Oracle USM to generate two different DIAMETER session identifiers. At the same time, the Oracle USM needs to maintain the AAR and STR associations with the original SIP session. To keep this relationship clear, the Oracle USM keeps the DIAMETER session identifier between the two media sessions the same for the two UAs.

The MBCD session identifier associated with the media session for the original SIP session is retained for the DIAMETER session identifier for the duration of the SIP session.

Accounting

There is no difference between an ACR record for a typical SIP session and one for a transcoded section, except that the direction of the forward codec (Acme-FlowType_FS1__F) and the reverse codec (Acme-FlowType_FS1__R). For both RADIUS and DIAMETER records, the IP address and port corresponding to the original two endpoints will be reflected; the T-SBC's details will not appear in the records.

Dynamic Sessions Agents for Home-Remote S-CSCF Liveliness

For IMS applications, you can configure your Oracle USM to create session agents dynamically for remote S-CSCFs on in-coming service routes. Dynamic session agents inherit properties of the static session agents with which they are associated, and the Oracle USM takes them out of service when they are deemed no longer responsive according to the liveliness mechanism you set.

Discovery

The Oracle USM, acting as a P-CSCF, can discover remote S-CSCFs using Service-Route header that returns with a 200 OK response from the registrar for a REGISTER request from the endpoint. The system takes the top Service Route from the response and uses it as the first hop in the egress route for the endpoint. Because the creation of dynamic session agents is based on the Service Route returned in the 200 OK, there is no impact to handling AoRs with multiple contacts.

In addition, the system stores the Service Route header data with the endpoint's registration cache. If the Service Route is an FQDN, a DNS look-up is used to provide the route to the S-CSCF.

Creation

For your Oracle USM to create session agents dynamically, you must enable the **create-dynamic-sa** in the global SIP configuration. This parameter defaults to **disabled**, meaning that no dynamic session agents will be created.

When you set the parameter to **enabled**, the Oracle USM decides whether or not to create dynamic session agents in the following ways. The system will create up to five (5) dynamic session agents.

- If the Service Route is an IP address, the Oracle USM attempts to find an exact match for that IP address against pre-existing session agents. If no exact match appears, the system will not create dynamic session agents.
- If the Service Route is an FQDN and matches an existing session agent, the Oracle USM assigns the remote S-CSCF to that session agent.
  Service Routes with FQDNs might not match any session agents. However, the Service Route's FQDN can match the DNS suffix of a wildcard session agent. When this wildcard matching occurs, the Oracle USM creates a new dynamic session agent with the original

wildcard session agent as its parent. The new dynamic session agent inherits all configuration properties of the parent session agent.

- If the Service Route is neither an IP address nor an FQDN and the system is unable to match any statically defined or wildcard session agents, then the Service Route is not associated with any session agents.

Property Inheritance

Because dynamic session agents are created on the basis of static or wildcarded session agents, you can think of them as children of the original--or parent--session agent. This relationship means that the child, dynamic session agent inherits the configuration properties of its parent, static or wildcard session agent.

These inherited configuration properties and their effects are:

- The FQDN's DNS resolution of the new dynamic session agent to IP addresses.
- Monitoring for liveliness via your configuration settings for the ping method and interval, transaction timeout, or OOS response. The dynamic session agent also inherits its parent's criteria for being taken out of service, with the exception that a dynamic session agent will not be taken out of service until it expires on its own. This gives the dynamic session agent time to return to service.
- The setting for **ping-all-address**, which when enabled causes new routes (internal session agents) fork and makes the dynamic session agent the parent. The system caps the limit of five routes (or internal session agents) per dynamic session agent. If the FQDN resolves to more than five IP addresses, the system only uses the first five to create routes (internal session agents), and then pings the internal session agents.
- Suppression of the heartbeat (or ping) of the IP address in the presence of traffic. If traffic to a specific IP address stops, then the Oracle USM resumes pinging within the time you set for the **ping-interval**.
- The setting for the **invalidate-registration** option. If the dynamic child session agent goes out of service, the corresponding registration cache entries of users that have the Service Route pointing to this session agent will be invalidated.
  This invalidation means that the next REGISTER request from that user will not receive a local response. Any other services this user requires will not use the Service Route information stored in its registration cache. Instead, the system will route it to the next hop as determined by other means, such as the local policy. At this time, the user would must be re-registered by the registrar, a process that might return a new Service Route to be updated in the registration cache.

Deletion

The Oracle USM needs to delete dynamic session agents no longer in use. But dynamic session agents should not be deleted too soon, in case they return to service. So, deletion occurs according to the process this section describes.

For each dynamically created session agent, the Oracle USM assigns and tracks the last registration expiry time. It determines this time by doubling the registered endpoint's core side expiry. Where X is the last registration expiry value and Y is the registered endpoint's core side expiry, the Oracle USM performs checks according to this criteria:

X > (2 * Y)

The the system updates the expiry time with the greater value. This way, whenever the dynamic session agent re-registers, it will have an updated expiry timer value.

Timeouts can also cause dynamic session agents to be deleted. Pings, status changes, transaction timeouts, DNS expiries and other system occurrences can trigger timeouts. When the Oracle USM detects that a timeout has occurred, the dynamic session agent is deleted.

How to Wildcard a Session Agent

You can create a wildcard session agent using the session agent's hostname parameter when configured with an FQDN.

To configure a wildcard session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are wildcarding and existing session agent, you have to select the configuration before you make changes.

4. **hostname**—To wildcard a session agent, you simply replace the value you want to wildcard with an asterisk (*). Also note that your value must lead with the asterisk (*), as in the following example.

   ```
   ORACLE(session-router)# hostname *xyz.com
   ORACLE(session-agent)#
   ```

5. Save and activate your configuration.

Enabling the Global SIP Configuration for Dynamic Session Agents

To use dynamic session agents for remote S-CSCFs on in-coming service routes, you need to set the create-dynamic-sa parameter in the global SIP configuration to enabled.

To configure the ping mode for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding this support to a pre-existing SIP configuration, you have to select the configuration before you make changes.

4. **create-dynamic-sa**—To support the creation of dynamic session agents for remote S-CSCFs on in-coming service routes, change this parameter from **disabled** (default) to **enabled**.
5. Save and activate your configuration.

Enhanced eSRVCC Call Continuity

As the LTE Evolved Packet Core (EPC) continues to expand, voice deployments (VoLTE) appear more and more, with the Oracle SBC and P-CSCF playing key roles. 3GPP standards that define how LTE communications take place also continue to evolve, identifying and solving critical issues to increase effectiveness and efficiency. One such issue is session continuity, keeping session transfers between LTE and existing 2G and 3G networks as seamless as possible. Single Radio Voice Call Continuity (eSRVCC) offers one solution for the session continuity issue.

In its role as the P-CSCF and IMS-GW, the Oracle USM can provide eSRVCC by acting as signaling and media anchor points to handover calls to 3G networks smoothly. These anchoring points are called Access Transfer Control Function (ATCF) and the Access Transfer Gateway (ATGW), both of which are logical additions to the Oracle USM's IMS support.

The behavior of these two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10. To align with Release 12, Requirement 4944 introduces the following two new elements to the functional design:

sip-feature-caps in session-router

The **sip-feature-caps** element lets the ATCF announce a feature capability in a message by transporting the information in the Feature-Caps header . The element has three parameters:

- **state** — The value "enabled" triggers the feature and adds the Feature-Caps header to messages. The default value is "disabled".
- **atcf-management-uri** — identifies the feature capability indicator that will be used to transport the ATCF management URI. Possible values are "management" and "psi". The default value is "management". When the value is "management" and the value of state is "enabled", the Feature-Caps header "g.3gpp.atcf-mgmt-uri" is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element. When the value is "psi" and the value of state is "enabled", the Feature-Caps header "g.3gpp.atcf-psi" is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element.
- **atcf-alerting** — The value "enabled" adds the Feature-Caps header to messages and turns on the alerting feature. The default value is "disabled".

When **state** is set to "enabled", **atcf-management-uri** is empty, and **atcf-alerting** is set to "disabled", then the Feature-Caps header is added with:

- g.3gpp.atcf : value is configured STN-SR in sip-config
- g.3gpp.atcf-mgmt-uri: the value is atcf-psi-dn in sip-config
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator

When **state** is set to “enabled”, **atcf-management-uri** is set to "psi", and **atcf-alerting** is set to "disabled", then the Feature-Caps header is added with:

- g.3gpp.atcf : value is configured STN-SR in sip-config
- g.3gpp.atcf-psi: the value is atcf-psi-dn in sip-config
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator

When **state** is set to “enabled”, **atcf-management-uri** is set to "management", and **atcf-alerting** is set to "disabled", then the Feature-Caps header is added with:

- g.3gpp.atcf : value is configured STN-SR in sip-config
- g.3gpp.atcf-mgmt-uri: the value is atcf-psi-dn in sip-config
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator

When **state** is set to “enabled”, **atcf-management-uri** is set to "management", and **atcf-alerting** is set to "enabled", then the Feature-Caps header is added with:

- g.3gpp.atcf : value is configured STN-SR in sip-config
- g.3gpp.atcf-mgmt-uri: the value is atcf-psi-dn in sip-config
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator
- g.3gpp.srvcc-alerting capability indicator

When **state** is set to “enabled”, **atcf-management-uri** is set to "psi", and **atcf-alerting** is set to "enabled", then the Feature-Caps header is added with:

- g.3gpp.atcf : value is configured STN-SR in sip-config
- g.3gpp.atcf-psi: the value is atcf-psi-dn in sip-config
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator
- g.3gpp.srvcc-alerting capability indicator

atcf-icsi-match in sip-interface

The **atcf-icsi-match** element lets the system check, on reception of an INVITE on an ingress sip-interface that has a configured ATCF and before applying any of the already implemented logic, whether the incoming INVITE includes the ICSI (Instantaneous Channel-State Information) of the requested service. The ATCF will be involved in the call flow when the configured ICSI value matches the ICSI value in the original INVITE; otherwise the handoff call will be rejected with code 404 (not found). The system looks for the ICSI string in the following headers:

- P-Preferred-Service
- P-Asserted-Service
- Feature-Caps (within the "g.3gpp.icsi-ref" feature-capability indicator)
- Accept-Contact (within the tag-value within the g.3gpp.icsi-ref media feature tag)

An example of the ICSI string in the P-Preferred-Service or P-Asserted-Service header is "urn:urn-7:3gpp-service.ims.icsi.mmtel". Examples of the ICSI string in the Feature-Caps or

Accept-Contact headers are "+g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel" and "g.3gpp.icsi-ref="urn%3Aurn-7%3A3gpp-service.ims.icsi.mmtel". The ATCF will be involved in the call flow only when the ICSI matches. If **atcf-icsi-match** is blank, the check is not done and the behavior remains the same as before.

Handsets and Session Continuity

Session continuity can be effected by handsets, which can be dual-mode (3G+WiFi and 3G +WiMAX, for example). Such a handset has two receivers or radios to initiate calls simultaneously. An LTE handset has only one receiver and is able to attach to a single LTE or 3G network at a given time.

The question of session continuity appears in 3GPP standards Release 8, 9, and 10—each building on the previous. Release 8 defines Single Radio Voice Call Continuity (SRVCC) as the mechanism for moving active voice sessions between LTE and existing 2G or 3G circuit networks. In moving over sessions such as these, it is key to keep latency as low as possible to increase the possibility of successful handovers.

Release 9, because of variable signaling latencies within the core network, could not guarantee smooth handovers to circuit networks. And though IMS provides great flexibility in locating application servers remotely from the UE, that flexibility actually increases the latency in signaling media changes to the access network. Due to the high total signaling latency and the difficulty of successfully coordinating handover timing between 3G and 4G call legs, the possibility of call drops increased.

Anchors for Signaling and Media

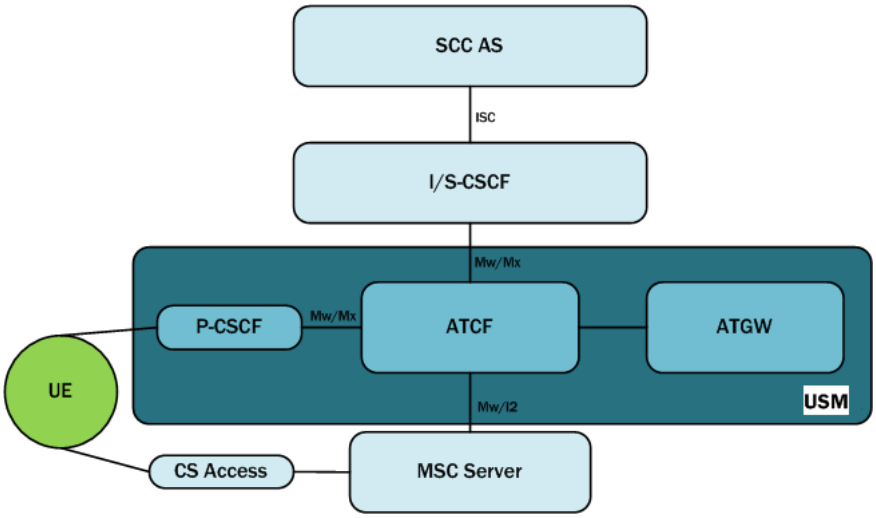
Release 10 addresses the latency concern by proposing these two logical entities, the anchoring points called the ATCF and ATGW. The Oracle USM can fulfill the tasks set to both entries:

- ATCF—A signaling anchor point co-located with the UE access network. The ATCF is responsible for:
  - Allocating the Session Transfer Number for Single Radio (STN-SR).
  - Instructing the ATGW to anchor the media path for originating and terminating sessions.
  - Tracking sessions (in alerting, active, or held states) so it can perform the access transfer of the selected session. Tracking this information allows the ATCF to support transferring the first session.
  - Performing Access Transfer and updating the ATGW with the new media path for the access call leg, without requiring updating from the remote leg.
  - After the access transfer, updating the Service Centralization and Continuity Application Server (SCC AS) that the transfer has taken place, ensuring that the Terminating Access Domain Selection (T-ADS) has updated information about the access currently used.
  - Handling failures during the access transfer.
  - Handling mid-call support for the access transfer using MSC server-assisted mid-call support.
- Access Transfer Gateway (ATGW)—A media anchor point co-located with the UE access network. Controlled by the ATCF, the ATGW anchors media both for the duration of a call and after the access transfer, based on the local configuration in the serving network.

Originating and terminating sessions are anchored in the ATCF and ATGW already during session set-up. For the first transferred session and the second established session, the SCC-AS provides session state information for the alerting, held, and conference states.

When a UE makes or receives a call, signaling and media are anchored at the ATCF and ATGW. At the point call handover point, the Visited-Mobile Switching Center (V-MSC) receives the handover message from Mobility Management Entity (MME, an EPC network element). The V-MSC then sends a call request to the local ATCF rather than sending the call request home or to the SCC-AS, as defined in 3GPP Release 8. Sending the call request to the ATCF reduces the number of hops required to initiate a media stream change to a new access network.

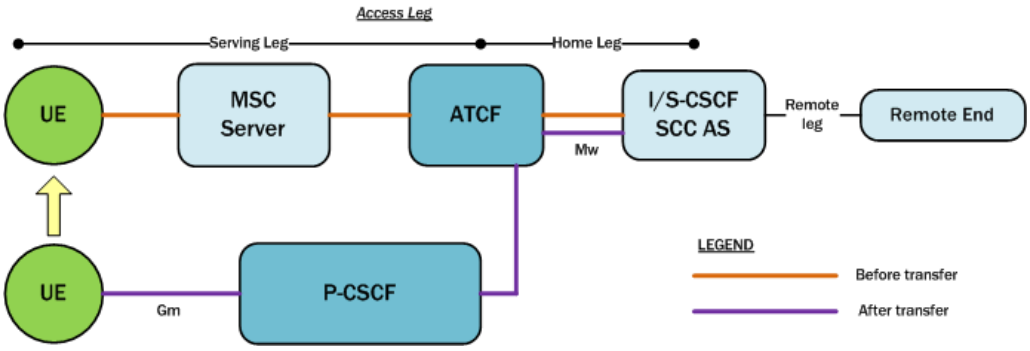
Acting as the ATGW, the Oracle USM can immediately affect media switchovers without further core signaling. In this role, the Oracle USM keeps RTP media continuity between endpoints using its Hide Media Update (HMU) functionality: The elements on the core side of the eSRVCC update will not see changes to SDP sources and destinations because the Synchronization Source identifiers (SSRCs) are masked. Media can thus flow directly to and from the 3G-attached MSC and onward to the UE with minimal interruption.



Note that if the MSC server-assisted mid-call feature is not supported or MSC server is not enhanced for ICS support, the interface between the MSC server and the ATCF will be Mw. If the MSC server-assisted mid-call feature is supported or the MSC server is enhanced for ICS, the interface between the MSC server and the ATCF will be I2.

Architectural View

This diagram is an architectural view of the control and user planes, depicting communication both before and after transfer. This depiction assumes the PGW and the P-CSCF are in the serving network and support IMS voice roaming (if not the home network). So, the ATCF resides in the serving networking (home network if not roaming). The access leg of the session is divided into the serving leg and home leg. In an actual network, there might be other server IMS nodes.



IMS Registration Details

This section discussed the IMS registration process when a UE attempts to register with the home network, but must do so using an ATCF/P-CSCF. Based on the operator policy and the home network's support for eSRVCC, the ATCF allocates an STN-SR to the session and includes itself in the signaling path for subsequent registration-related messaging. To understand whether or not the home network supports eSRVCC, the ATCF uses service-level agreements and can also determine if eSRVCC is activated in the SCC AS by the reception of a C-MSISDN/ATU-SI during session start-up.

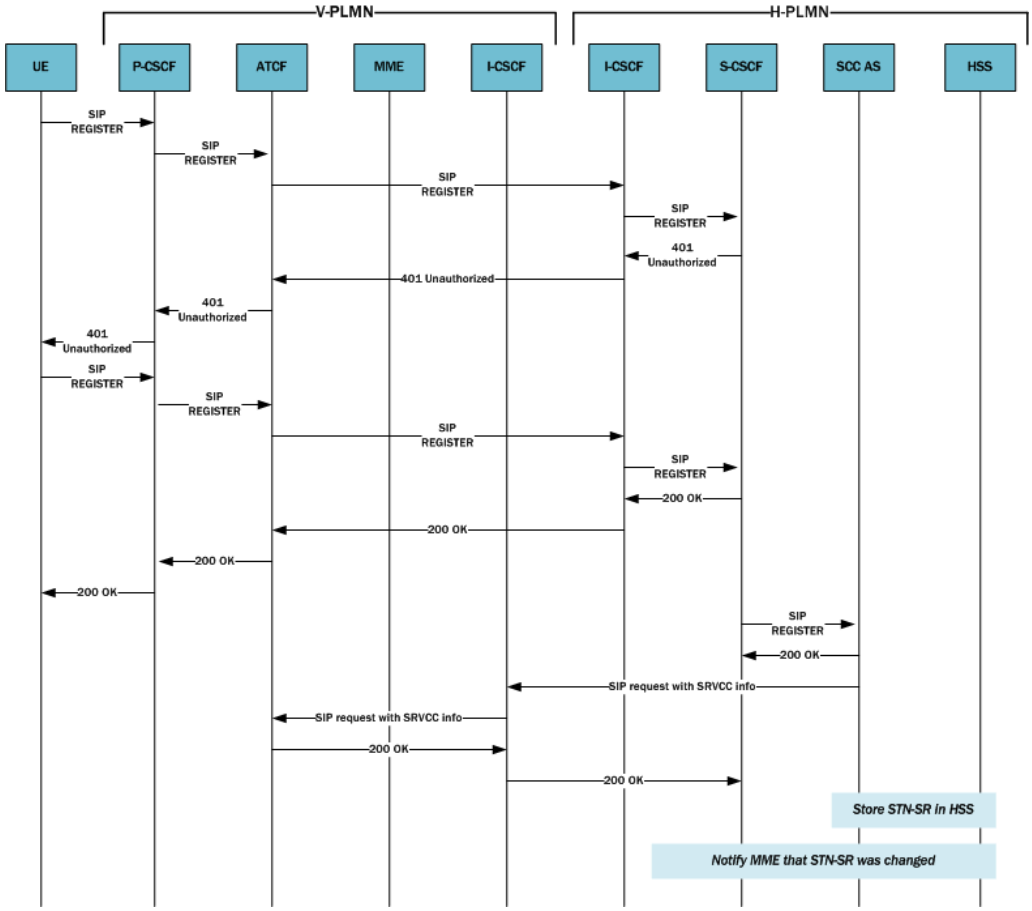
If the ATCF generates an STN-SR, it includes the STN-SR in requests it forwards to the I/S-CSCF. The path and route information for the SIP REGISTER request sent to the S-CSCF is:

- Path—ATCF URI for terminating requests (uniquely identifies registration or registration flow), followed by P-CSCF URI for terminating requests.
  The header field containing the ATCF URI for terminating requests contains the g. 3gpp.atcf media feature tag (indicating this URI supports the ATCF role). This tag contains the STN-SR and the ATCF PSI, showing this URI can receive SIP message requests with SRVCC-related information.
- Route—URI of the entry point of the UE's home network.

The ATCF tracks existing registrations for the UEs it has served. Each registration is identified by the P-CSCF path URI. The following information remains in the ATCF's registration cache: the S-CSCF service route URI, the ATU-STI, and the C-MSISDN. When a UE's registration expires or it is de-registered, the ATCF can remove any SRVCC information bound to the registration.

This ladder diagram shows the IMS registration process between these entities:

- The Visited Public Land Mobile Network (V-PLMN)—The network a mobile subscriber uses when that subscriber is roaming.
- The Home Public Land Mobile Network (H-PLMN)—The mobile subscriber's home network.



SIP Register Request UE to ATCF

The following is an example of the SIP REGISTER request the UE sends to the ATCF/ P-CSCF.

```
REGISTER sip:home1.net SIP/2.0
Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:eee];comp=sigcomp;branch=z9hG4bKnasiuen8
Max-Forwards: 70
P-Access-Network-Info: 3GPP-UTRAN-TDD; utran-cell-id-3gpp=234151D0FCE11
From: <sip:user1_public1@home1.net>;tag=2hiue
To: <sip:user1_public1@home1.net>
Contact: <sip:[5555::aaa:bbb:ccc:eee];comp=sigcomp>;+sip.instance="<urn:gsma:imei:90420156-
   025763-0>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Call-ID: E05133BD26DD
Authorization: Digest username="user1_private@home1.net", realm="registrar.home1.net", nonce="",
   uri="sip:home1.net", response=""
Security-Client: ipsec-3gpp; alg=hmac-sha-1-96; spi-c=23456789; spi-s=12345678; port-c=1234;
   port-s=5678
Require: sec-agree
Proxy-Require: sec-agree
CSeq: 1 REGISTER
Supported: path, gruu
Content-Length: 0
```

SIP Register Request ATCF to S-CSCF

The following is an example of the SIP REGISTER request the ATCF sends to the S-CSCF.

```
REGISTER sip:home1.net SIP/2.0
Path: <sip:termsdgfdfwe@actf.visited2.net>;+g.3gpp.atcf="<tel:+1-237-888-9999>";+g.3gpp.atcf-
   psi="<sip:actf.visited2.net>",<sip:aga2gfgf@pcscf1.visited2.net:5070;ob>
Route: <sip:icscf.home1.net;lr>
P-Visited-Network-ID:
P-Charging-Vector:
Via: SIP/2.0/UDP actf.visited2.net:5060;branch=z9hG4bKnas5889; SIP/2.0/UDP
   pcscf1.visited2.net:5060;branch=z9hG4bKnas56565, SIP/2.0/UDP
   [5555::aaa:bbb:ccc:eee];comp=sigcomp;branch=z9hG4bKnasiuen8;rport=5060;received=5555::aaa:bbb:
   ccc:eee
Max-Forwards: 68
P-Access-Network-Info:
From:
To:
Contact:
Call-ID:
Authorization:
Require:
Proxy-Require:
CSeq:
Supported:
Content-Length:
```

SIP 200 OK from S-CSCF

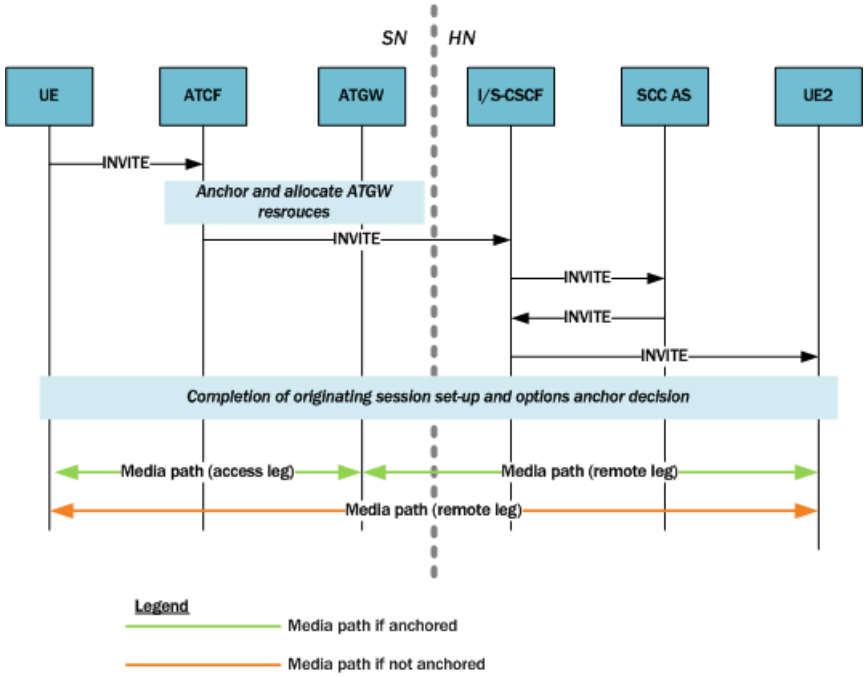
The following is an example of the SIP 200 OK from the S-CSCF.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP icscf1_p.home1.net;branch=z9hG4bK351g45.1, SIP/2.0/UDP
     pcscf1.visited1.net;branch=z9hG4bK240f34.1, SIP/2.0/UDP
     [5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp;branch=z9hG4bKnashds7
Path: <sip:term@pcscf1.visited1.net;lr;ob>
Service-Route: <sip:orig@scscf1.home1.net;lr>
From:
To:
Call-ID:
Contact: <sip:[5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp>;
   pub-gruu=" sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
   ;temp-gruu="sip:tgruu.7hs==jd7vnzga5w7fajsc7-ajd6fabz0f8g5@example.com;gr"
   ;+sip.instance="< urn:gsma:imei:90420156-025763-0 >"+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-
   service.ims.icsi.mmtel";+g.3gpp.ics="principal";+g.3gpp.accesstype="cellular1"
   ;expires=600000;+g.3gpp.iut-controller
CSeq:
Supported: path, outbound
Require: outbound
Date: Wed, 11 July 2001 08:49:37 GMT
P-Associated-URI: <sip:user1_public2@home1.net>, <sip:user1_public3@home1.net>, <sip:+1-212-555-
   1111@home1.net;user=phone>
Content-Length:
```

Originating Sessions for SRVCC with ATCF

For initial SIP requests, the ATCF distinguishes SIP INVITE requests with the ATCF URI for originating requests in the topmost Route header field. And when receiving such an originating SIP INVITE request, the ATCF will do the following prior to forwarding it:

- Insert the Record-Route header field with its own SIP URI.
- If the latest SRVCC information received for a session contains a C-MSISDN and ATU-STI:
  - The ATCF will associate the session being established with the C-MSISDN and ATU-STI bound to the registration.
  - The ATCF will replace the SDP offer in the originating SIP INVITE with updated SDP the ATGW provides. Replacement occurs if the originating SIP INVITE contains SDP and a determination to anchor media has been made (according to the operator policy as specified in the 3GPP standard).



SIP INVITE for SRVCC Using the ATCF

The following is an example of the SIP INVITE the UE sends to the ATCF/P-CSCF.

```
INVITE tel:+1-212-555-2222 SIP/2.0
Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:ddd]:1357;comp=sigcomp;branch=z9hG4bKnashds7
Max-Forwards: 70
Route: <sip:pcscf1.visited2.net:7531;lr;comp=sigcomp>, <sip:orig@scscf1.home1.net;lr>
P-Preferred-Identity: "John Doe" <sip:user1_public1@home1.net>
P-Preferred-Service: urn:urn-7:3gpp-service.ims.icsi.mmtel
P-Access-Network-Info: 3GPP-UTRAN-TDD; utran-cell-id-3gpp=234151D0FCE11
Privacy: none
From: <sip:user1_public1@home1.net>;tag=171828
To: <tel:+1-212-555-2222>
Call-ID: cb03a0s09a2sdfglkj490333
Cseq: 127 INVITE
Require: sec-agree
Supported: precondition, 100rel, gruu
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi-c=98765432; spi-s=87654321; port-
   c=8642; port-s=7531
Contact: <sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-
   00a0c91e6bf6;comp=sigcomp>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Accept-Contact: *;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Content-Type: application/sdp
Content-Length: (…)

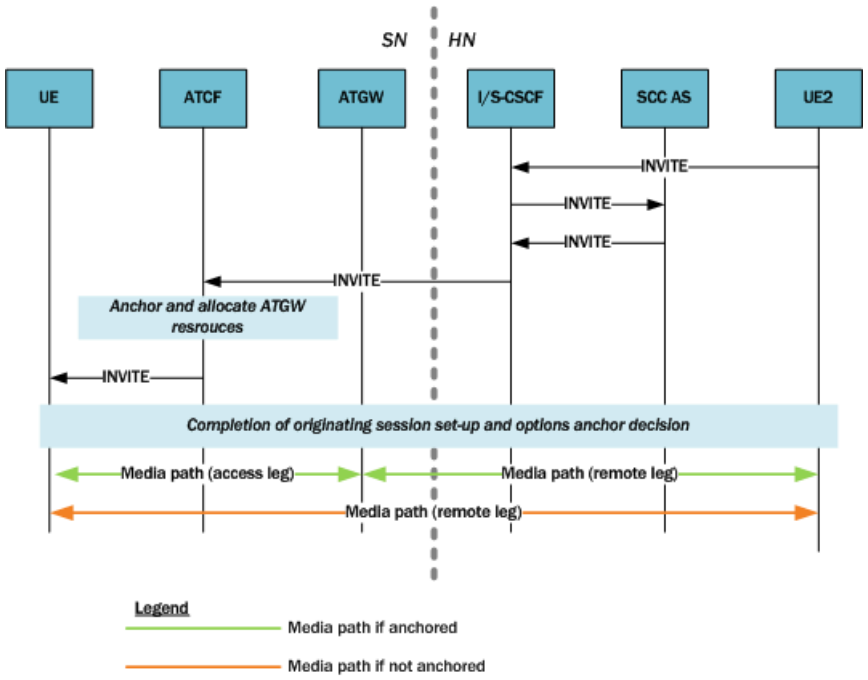
v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
s=-
c=IN IP6 5555::aaa:bbb:ccc:ddd
t=0 0
m=audio 3456 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; maxframes=2
a=rtpmap:96 telephone-event
```

Terminating Sessions for SRVCC with ATCF

For initial SIP requests, the ATCF identifies SIP INVITE requests with the ATCF URI for terminating requests in the topmost Route header field. These are called terminating SIP INVITE requests.

When it receives a terminating SIP INVITE, the ATCF does the following if the INVITE has a Record-Route header field with the g.3gpp.srvcc media feature tag:

- The ATCF inserts a Record-Route header field with its own SIP URI
- If the latest SRVCC information received for a session contains a C-MSISDN and ATU-STI,
  - The ATCF associates the session being established with the C-MSISDN and ATU-STI bound to the registration, and
  - The ATCF replaces the SDP offer in the originating SIP INVITE with updated SDP provided by the ATGW. Replacement occurs if the terminating SIP INVITE contains an SDP offer and a determination to anchor media has been made (according to the operator policy as specified in the 3GPP standard).



SIP INVITE from UE2 ATCF

The following is an example of the SIP INVITE UE2 sends to the ATCF/P-CSCF.

```
INVITE < sip:user1_public1@home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6> SIP/2.0
Via: SIP/2.0/UDP sccas1.home1.net;branch=z9hG4bKnas34r5
Max-Forwards: 67
Route: <sip:scscf1.home1.net:lr>
P-Asserted-Identity: <tel: +1-237-555-2222>
P-Charging-Function-Addresses: ccf=[5555::b99:c88:d77:e66]; ccf=[5555::a55:b44:c33:d22];
   ecf=[5555::1ff:2ee:3dd:4ee]; ecf=[5555::6aa:7bb:8cc:9dd]
P-Charging-Vector: icid-value="AyretyU0dm+6O2IrT5tAFrbHLso=023551024"; orig-ioi="type3home1.net"
P-Access-Network-Info:
Privacy: none
From: <tel: +1-237-555-2222; gr=hdg7777ad7aflzig8sf7>;tag=171828
To: <tel:+1-237-555-1111>
Call-ID: cb03a0s09a2sdfglkj490333
Cseq: 127 INVITE
Supported: 100rel, precondition
Require: sec-agree
Proxy-Require: sec-agree
Security-Verify: ipsec-3gpp; q=0.1; alg=hmac-sha-1-96; spi=87654321; port=7531
Contact: < sip:user2_public1@home2.net;gr=urn:uuid:2ad8950e-48a5-4a74-8d99-
   ad76cc7fc74>;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER, MESSAGE
Accept: application/sdp, application/3gpp-ims+xml
Content-Type: application/sdp
Content-Length: (…)

v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
s=
c=IN IP6 5555::aaa:bbb:ccc:ddd
t=0 0
m=audio 3456 RTP/AVP 97 96
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
m=message 0 TCP/MSRP 98
a=accept-types:text/plain
```

TS 24.237 Proposed Changes

The Oracle USM implements a proposed change in the processing of failed or cancelled SRVCC sessions. The new processing model has been presented to the 3GPP for inclusion in TS 24.237, IP Multimedia (IM) Core Network (CN) subsystem IP Multimedia Subsystem (IMS) service continuity; Stage 3.

Sections 12.2.4.13, 12.3.3.1, and 12.3.3.1A of 3GPP TS 24.237, IP Multimedia (IM) Core Network (CN) subsystem IP Multimedia Subsystem (IMS) service continuity; Stage 3, describe procedures in response to SRVCC handover cancellation. These procedures allow the UE to generate a e-INVITE request with the Reason header field containing a value of “SIP”, a “cause” parameter set to a value of “487” (Request Terminated), and with reason-text parameter set to a value of either “handover cancelled” or “failure to transition to CS domain”.

SCC AS processing, which does nor release the original access leg, is defined in Section 12.3.3.1 and subclause 12.3.3.1A. Section 13.3.1 describes the handling of subsequent UPDATE requests. There are, however, no defined procedures at the ATCF response to this scenario with the result that the ATCF fails to reconfigure the ATGW to reconnect the bearer in LTE.

Oracle Corporation, in conjunction with other interested parties has proposed the addition of a new Section 12.7.2.3.3 to TS 24.237. This section requires the following ACTF behavior.

Requirement 1: When the ATCF receives either a

- SIP BYE request on the Source Access Leg containing a Reason header field containing a SIP 503 (Service Unavailable) response code, that is terminating an established dialog or an early dialog on the Source Access Leg
- SIP CANCEL request on the Source Access Leg with the Reason header field containing a SIP 503 (Service Unavailable) response code then, that is terminating an early dialog on the Source Access Leg originated by the SC UE, or
- SIP 503 (Service Unavailable) response on the Source Access Leg, that is terminating an early dialog on the Source Access Leg terminating at the SC UE

Then -- The ATCF shall retain session state information and ATGW resources associated with the session until either it receives a SIP INVITE request due to STN-SR, or a specified time period elapses (default value is 8 seconds).

The session remains recognizable for SRVCC access transfer as described in Section 12.7.2.1.

The SIP BYE request is forwarded to the SCC AS, which also delays release of the session, as described in Section 12.3.3.2.

Requirement 2: If the transferable session set determined in Section 12.7.2.1 does not contain any sessions and the identity in the P-Asserted-Identity header field is a C-MSISDN that is not bound to a registration path in the ATCF, the ATCF shall respond with a SIP 404 (Not Found) response.

Requirement 3: When the ATCF receives a SIP re-INVITE request containing Reason header field containing protocol “SIP” and reason parameter “cause” with value “487” on the original source access leg,

- after having initiated an access transfer that was triggered by a SIP INVITE request due to STN-SR, and
- the SIP INVITE request due to ATU-STI transaction is not yet completed

Then -- The ATCF shall wait until this transaction has completed and then continue with the steps described in Requirement 4.

Requirement 4: When the ATCF receives a SIP re-INVITE request(s) containing protocol “SIP” and reason parameter “cause” with value “487” after having performed an access transfer that was triggered by a SIP INVITE request due to STN-SR,

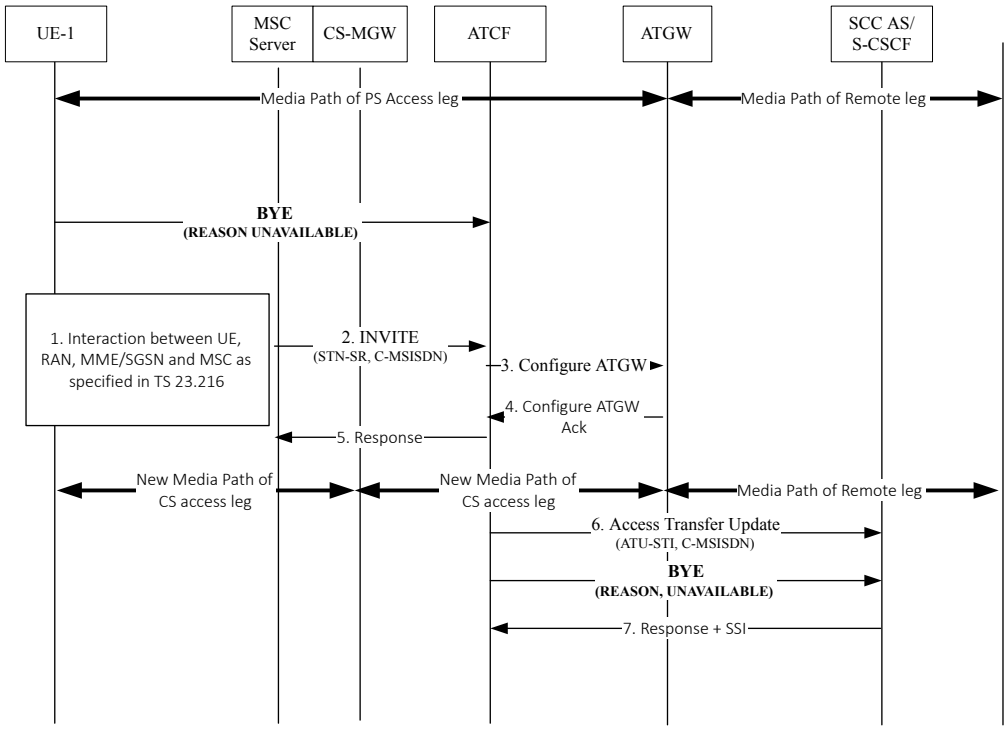
Then -- The ATCF shall act as B2BUA as described in Section 5.6 and shall.

1. Interact with ATGW to provide information needed in the procedures below and to request ATGW to start forwarding the media from the remote UE to the local UE. The details of interaction between ATCF and ATGW are out of scope of this document.
2. Send a SIP 200 (OK) response to the received SIP re-INVITE request. The SIP 200 (OK) response contains the SDP answer that includes the ATGW ports and the IP addresses as provided by the ATGW and the media used on the original source access leg as before the access transfer; and
3. Forward the received reINVITE with the Reason header intact to the SCC AS on the existing source dialog with the SDP offer containing the ATGW IP addresses and ports towards the remote UE as provided by the ATGW.

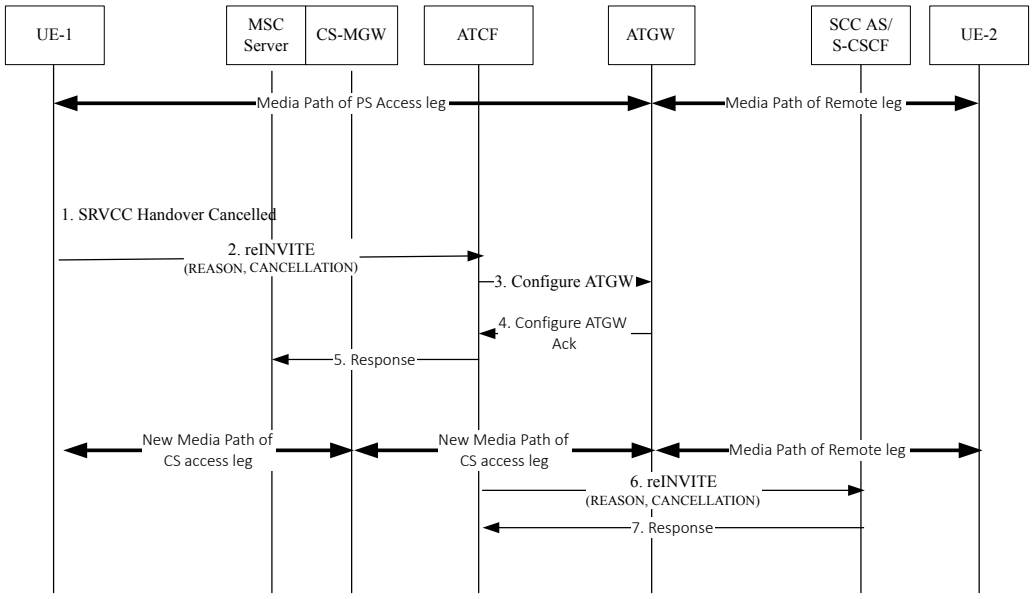
This proposed behavior, which requires no user configuration) is implemented in Version S-CX7.2.0 and later releases.

Related call flows are shown below.

BYE before handover INVITE



Cancellation after handover complete



SRVCC PS-CS Access Transfer

This section describes various scenarios for transferring sessions between Packet-Switched (PS) and Circuit-Switched (CS) networks, providing details about the following cases:

- Mobile Switching Center (MSC) server-assisted mid-call feature supported by the SCC AS
- Call flows for the MSC server-assisted mid-call feature supported by the SCC AS
- Failure cases
- Confirmed dialog

- Early dialog

For PS-CS transfers in SRVCC, the ATCF performs several actions based on the STN-SR when it first receives the SIP INVITE. First, the ATCF determines the set of transferrable sessions. It defines this set as the SRVCC-transferrable sessions that are associated with a C-MSISDN matching the URI in the P-Asserted-Identity header field in the INVITE. The ATCF compares the INVITE to its set of transferrable sessions to see if the INVITE is part of the set; if so, its responds with 2xx response for the initial INVITE.

Active session transfers require the ATCF to act as a back-to-back user agent (B2BUA) if the session undergoing transfer has media anchored at the ATGW. In this case, the ATCF responds with a 200 OK to the INVITE; the OK carries an SDP answer with ATGW IP address and port information. Then the ATCF updates the following information and sends the INVITE to the remote UE:

- Original SDP offer is replaced with an SDP offer containing media information currently in use with the ATGW IP address and port information.
- The Request-URI with the ATU-SI associated with the session being transferred is inserted.
- The Target-Dialog header field with the dialog identifier of the session being transferred is inserted.

When the ATCF receives an SDP answer and if media is anchored at the ATGW for the session being transferred, the ATCF directs the ATGW to start sending and receiving media to/from the remote UE according to the newly received answer.

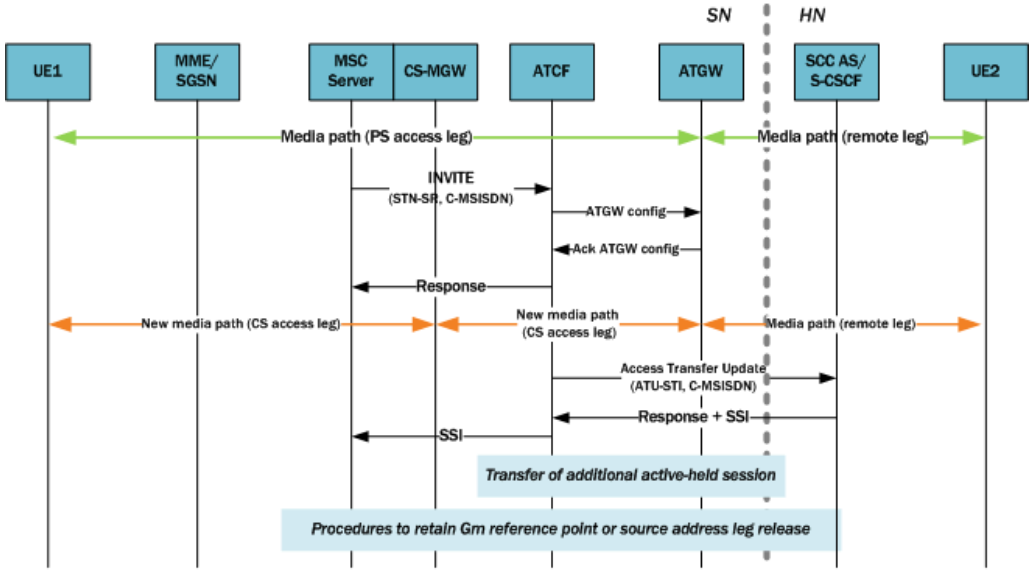
If it receives a BYE with a 503 Service Unavailable header on the source access leg, the ATCF retains session state information and ATGW resources associated with the session until either:

- The ATCF receives an INVITE request because of STN-SR, or
- An operator-defined time period elapses. The default value for the operator-defined time period is eight (8) seconds.

Thus, the session remains recognizable for SRVCC transfer for a time. Release of the session is also delayed by the ATCF's forwarding the BYE to the SCC AS. If the transferrable session cannot progress, the ATCF responds with a 404 Not Found. For transfer cases when only hold or alerting sessions exist and the session in question is an early dialog or a confirmed dialog with inactive media, the ATCF replaces the Request-URI received in the original INVITE with the ATU-STI associated with a session in the transferrable set before forwarding the request; this proxy-role behavior complies with the 3GPP standard.

MSC Server-Assisted Mid-Call Feature Supported by CSS AS

The following scenario assumes an active session between UA1 and UA2, and that UA1 attaches to the CS domain. It is also assumed that the CS-MGW supports the codecs used for LTE voice calls, minimizing the likelihood that the ATCF will need to instruct the ATGW to insert codecs.



The MSC server initiates a transfer and (if supported) indicates its support for the MSC server-assisted mid-call feature. The MSC server also provides all voice codecs it supports in the Access Transfer message.

The following shows a sample INVITE from the MSC server to the ATCF. Note the following:

- The Request-URI header contains the STN-SR, as routed to the ATCF.
- The P-Asserted-Identity header reflects the C-MSISDN of the UE being served.
- The From header reflects the C-MSISDN of the serving UE.
- The SDP contains a pre-configured set of codecs the MGW supports.

```
INVITE tel: +1-237-555-3333 SIP/2.0
Via: SIP/2.0/UDP msc1.visit1.net;branch=z9hG4bk731b87
Max-Forwards: 70
P-Asserted-Identity: <tel:+1-237-555-2222>
P-Charging-Vector: icid-value="AyretyU0dm+6O2IrT5tAFrbHLso=023551024";orig-ioi=visit1.net
Privacy: none
From: <tel:+1-237-555-1111>;tag=171828
To: <tel: +1-237-555-3333>
Call-ID: cb03a0s09a2sdfglkj490334
Cseq: 127 INVITE
Supported: 100rel, precondition, gruu
Accept-Contact: *;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.mmtel
Contact: <sip:msc1.home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>;+g.3gpp.icsi-
 ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER
Content-Type: application/sdp
Content-Length: (…)

v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:eee
s=
c=IN IP6 5555::aaa:bbb:ccc:eee
t=0 0
m=audio 3456 RTP/AVP 97 96
a=tcap:1 RTP/AVPF
a=pcfg:1 t=1
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
```

Upon receiving the Access Transfer message, the ATCF identifies the correct anchored session and proceeds with the transfer of the most recently active session. Using a Configure ATGW

message, the ATCF replaces the existing PS access leg media path information with new CS access leg media path information for the ATGW. However, the ATCF might instruct the ATGW to continue using the local port of the PS access leg media path. Once the ATGW acknowledges the ATCF's Configure message, the ATCF sends the MSC server an Access Transfer response and the media path is moved to the CS when receiving SDP information.

Voice break interruption starts either when media moves to the CS MGW (controlled by the MSC server enhanced for SRVCC) or when the UE relocates to the target—whichever comes first. When the UE tunes to the target or media switches to the CS MGW (whichever is last), the voice break interruption ends. The assumption is that media is switched to the CS MGW during the time to UE tunes to the target.

After receiving the Access Transfer message, the ATCF re-establishes communication with the SCC AS and updates the SCC AS. When the MSC server supports the mid-call feature, the ATCF also communicates this fact to the SCC AS. The ATU message initiates a new dialog between the ATCF and SCC AS, a dialog the CSS AS associates with the old dialog using the C-MSISDN. This new dialog is necessary because it replaces the old dialog set up over PS access, thereby assuring that if the PS user registration expires the new home leg will not be released or even affected.

The following shows a sample INVITE request the ATCF sends to the S-CSCF. Note the following:

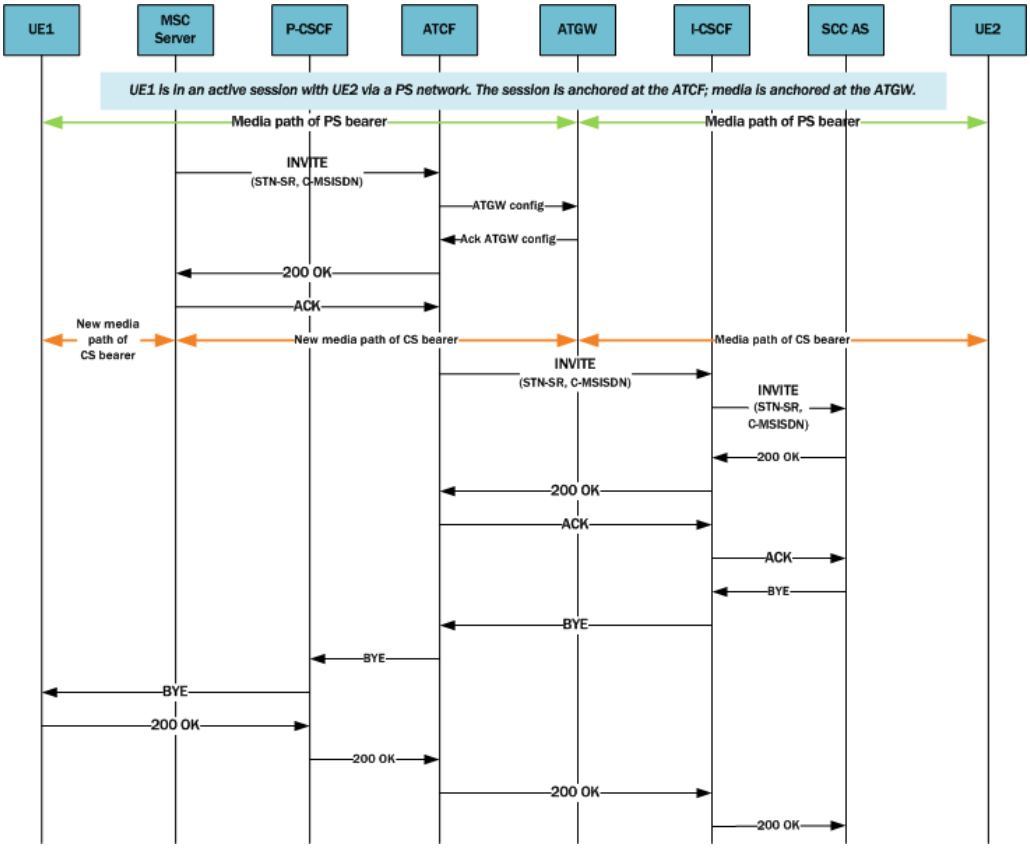
- The Request-URI header contains the ATU-STI, as routed to the SCC AS.
- The Target dialog header specifies the existing dialog is associated with this request.
- The P-Asserted-Identity header reflects the C-MSISDN of the UE being served.
- The From header reflects the C-MSISDN of the serving UE.
- The SDP reflects the media information at the ATGW.

```
INVITE sip:AUT-STI1@sccas.home1.net SIP/2.0
Via: SIP/2.0/UDP msc1.visit1.net;branch=z9hG4bk731b87
Max-Forwards: 70
P-Asserted-Identity: <tel:+1-237-555-2222>
P-Charging-Vector: icid-value="AyretyU0dm+6O2IrT5tAFrbHLso=023551024";orig-ioi=visit1.net
Privacy: none
From: <tel:+1-237-555-9999>;tag=171828
To: <tel: +1-237-555-4444>
Call-ID: cb03a0s09a2sdfglkj490334
Cseq: 127 INVITE
Supported: 100rel, precondition, gruu
Target-Dialog: me03a0s09a2sdfgjkl491777; to-tag=774321; from-tag=64727891
Accept-Contact: *;+g.3gpp.icsi-ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
P-Asserted-Service: urn:urn-7:3gpp-service.ims.icsi.mmtel
Contact: <sip:msc1.home1.net;gr=urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6>;+g.3gpp.icsi-
 ref="urn%3Aurn-7%3gpp-service.ims.icsi.mmtel"
Allow: INVITE, ACK, CANCEL, BYE, PRACK, UPDATE, REFER
Content-Type: application/sdp
Content-Length: (…)

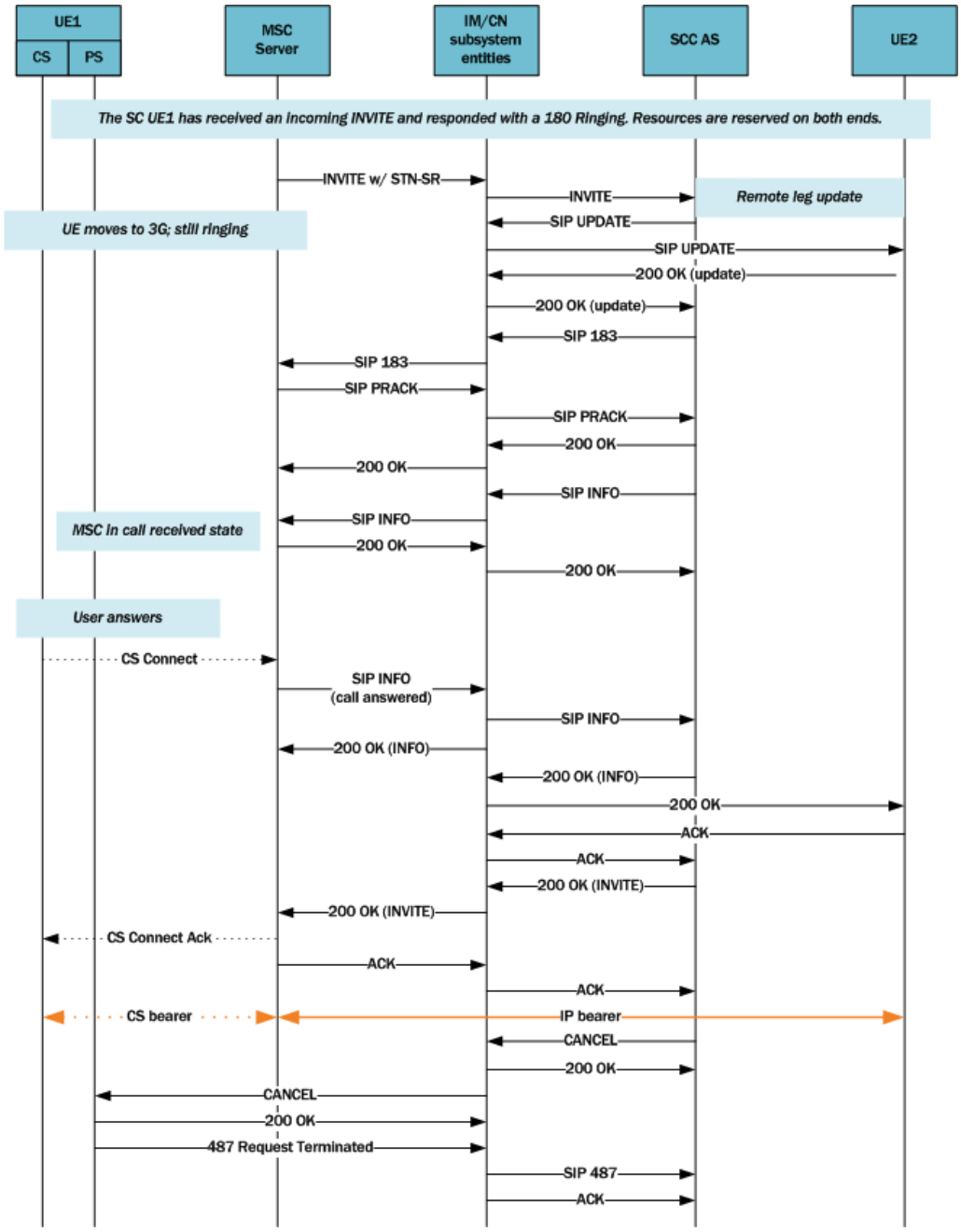
v=0
o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ggg
s=
c=IN IP6 5555::aaa:bbb:ccc:ggg
t=0 0
m=audio 3456 RTP/AVP 97 96
a=tcap:1 RTP/AVPF
a=pcfg:1 t=1
b=AS:25.4
a=curr:qos local sendrecv
a=curr:qos remote none
a=des:qos mandatory local sendrecv
a=des:qos none remote sendrecv
a=rtpmap:97 AMR
a=fmtp:97 mode-set=0,2,5,7; mode-change-period=2
a=rtpmap:96 telephone-event
a=maxptime:20
```

The SCC AS sends the ATCF a confirmation, including the session state information (SSI) if the SCC AS and the MSC server support the mid-call feature. The MSC server receives the SSI from the ATCF; the access leg for the control has moved to the CS access. When MSC server receives SSI for more multiple active or inactive speech sessions, it initiates transfer directed at the SSC AS for the additional sessions.

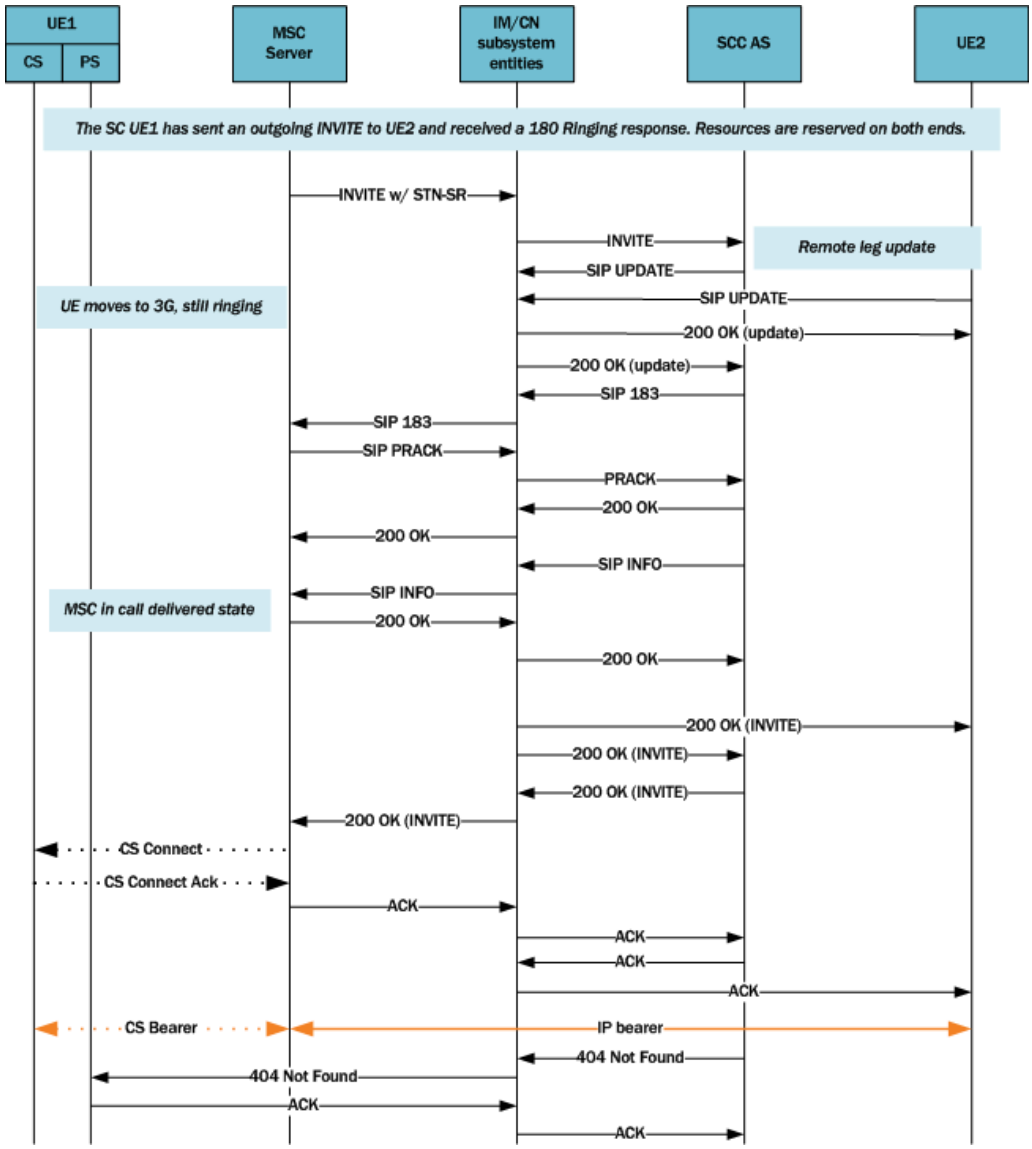
The following diagram shows a call session in the active phase for MSC server assisted mid-call feature when supported by the CSS AS.



This diagram shows an incoming call session during the alerting phase.



This diagram shows an outgoing call session during the alerting phase.



Failure and Cancellation

For cases of failure and/or cancellation, the ATCF behaves in accordance with TS 23.216 [3], Section 8:

- In the case of session failure before the MSC server initiates session transfer, the standard failover procedures (as defined in TS 23.401 [2])apply and no further action is required by the UE.
- In the case of session failure after the UE receives the handover command, the UE attempts to return to the Universal Mobile Telecommunications System (UTRAN) or evolved Universal Mobile Telecommunications System (eUTRAN). The UE initiates signaling to transfer the session back, which the ATCF handles if the ATCF recognizes the session transfer.
- In the case when handover is cancelled, the UE—having received the handover cancellation message—begins the re-establishment procedure as though it needed to

transfer the session to the eUTRAN/UTRAN. If he ATCF identifies this as a transfer session, it handles the session transfer back to the eUTRAN/UTRAN.

Confirmed Dialogs

When an SC UE is engaged in one or more ongoing IMS sessions, it will send a re-INVITE containing:

- An SDP offer, including the media characteristics used in the existing dialog, and
- A Reason header field containing the protocol SIP and reason parameters cause with the value 487 (as specified in RFC 3326 [57]), with reason text reading either Handover cancelled or Failure to transition to CS domain.

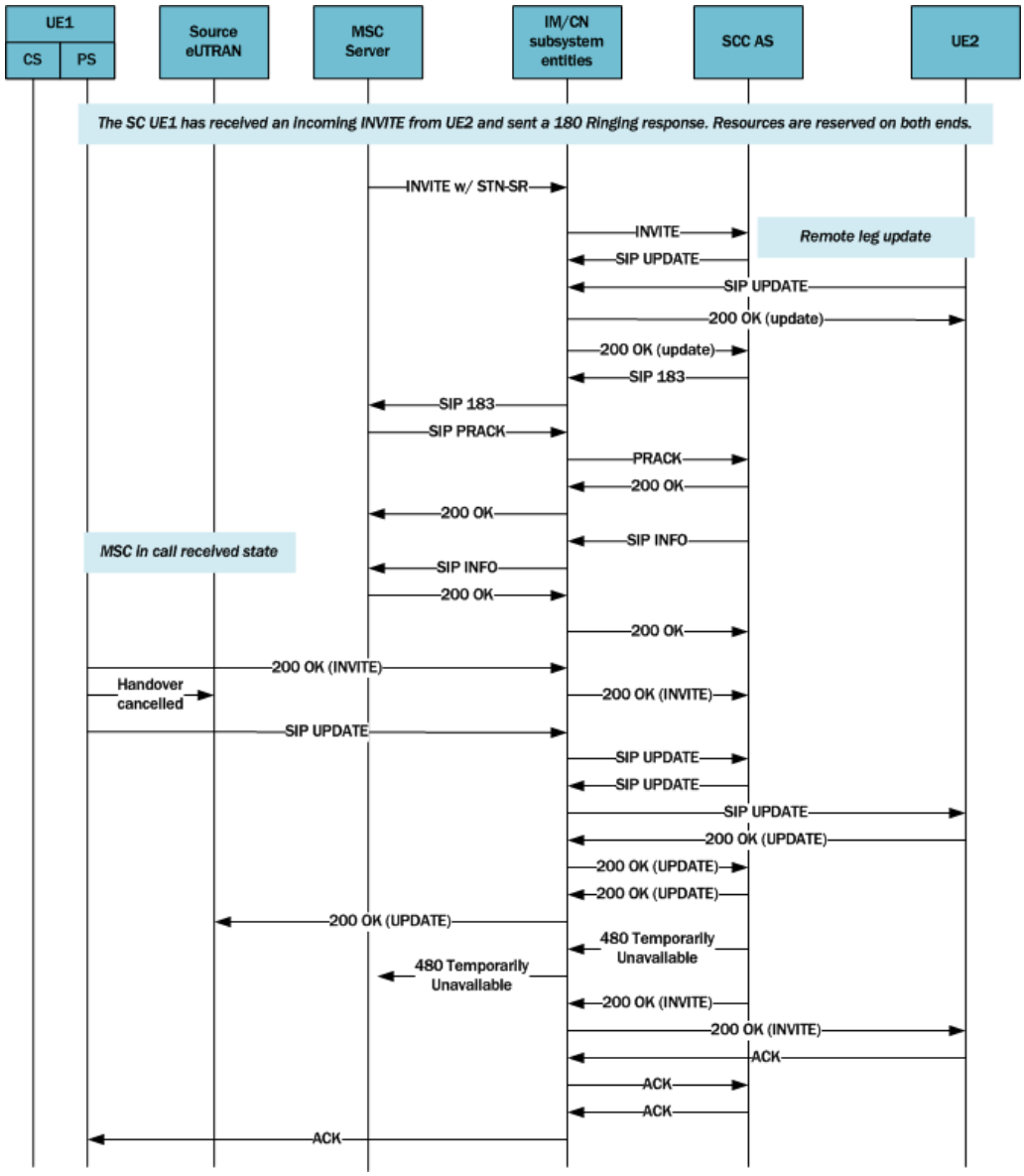
In all other ways, confirmed dialogs behave in accordance with TS 24.229 [2].

Early Dialogs

If the SC UE engages in a session which is in early dialog state, it will send a SIP UPDATE containing:

- An SDP offer, including the media characteristics used in the existing dialog, and
- A Reason header field containing the protocol SIP and reason parameters cause with the value 487 (as specified in RFC 3326 [57]), with reason text reading either Handover cancelled or Failure to transition to CS domain.

In all other ways, confirmed dialogs behave in accordance with TS 24.229 [2].



SRVCC Handover Support in Alerting Phase

Oracle now supports handovers between Packet-Switched (PS) and Circuit-Switched(CS) networks for calls in an alerting phase; that is, a 180 ringing response for the initial INVITE has been sent or received and the SIP final response has not been sent or received.

The behavior of the two anchoring points, ATCF and ATGW, is defined by 3GPP in Release 12 of Technical Specification TS 24.237. Oracle Communications developed these functional entities based on the initial version of TS 24.237 Release 10, and has added the **sip-feature-caps** configuration element to align with Release 12. The element has three parameters:

- **state** — The value "enabled" triggers the feature and adds the Feature-Caps header to messages. The default value is "disabled".
- **atcf-management-uri** — identifies the feature capability indicator that will be used to transport the ATCF management URI. Possible values are "management" and "psi". The default value is "management". When the value is "management" and the value of state is

"enabled", the Feature-Caps header “g.3gpp.atcf-mgmt-uri” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element. When the value is "psi" and the value of state is "enabled", the Feature-Caps header “g.3gpp.atcf-psi” is added and the value is the value of **atcf-psi-dn** in the **sip-config** configuration element.

- **atcf-alerting** — The value "enabled" adds the Feature-Caps header to messages and turns on the alerting feature. The default value is "disabled".

To ensure that calls in an alerting phase are transferred between PS and CS networks, set the value of **state** to “enabled”, set the value of **atcf-management-uri** to "management", and set the value of **atcf-alerting** to "enabled". The Feature-Caps header is then added with:

- g.3gpp.atcf : value is configured STN-SR in **sip-config**
- g.3gpp.atcf-mgmt-uri: the value is **atcf-psi-dn** in **sip-config**
- g.3gpp.atcf-path: value is the ATCF URI for terminating requests
- g.3gpp.mid-call capability indicator
- g.3gpp.srvcc-alerting capability indicator

For more information, refer to the SIP Feature Capabilities section.

Accounting

eSRVCC calls involve two SIP sessions, the accounting records for which need clear association. To correlate the records, the Oracle USM performs these actions for eSRVCC calls:

- When it detects a session handover, the system generates an Interim accounting record for the initial SIP session. This way, the records provide the exact time of the handover (i.e., the timestamp of the Interim record).
- The Oracle specific AVP Generic-ID will appear in Start, Interim, and Stop records for the handover SIP session. The Generic-ID contains the Call ID of the initial SIP session, which helps to correlate the two SIP sessions.
  - For RADIUS accounting records, refer to Oracle specific VSA 40.
  - For DIAMETER accounting records, refer to Oracle specific VSA 30.
- The Stop record for the initial SIP session will not have media flow information because the media session is considered part of the handover SIP session.
- If you are using QoS, the Stop records for the handover SIP session reflects the cumulative QoS statistics for both the initial SIP session and the handover SIP session.

External Bandwidth Management

Is you are using the Oracle USM's external bandwidth management for eSRVCC calls, note these considerations:

- At the time of handover and if the new handover realm has the same external policy server configured as the initial SIP session, the Rx will continue seamlessly with the same DIAMETER session identifier directed toward the policy server. From the perspective of the policy server, the same session is simply continuing.
- At the time of handover and if the new handover realm does not have an external policy server configured, policy server services will stop.

ATCF Configuration

ATCF functionality requires configuration in the **sip-config** and **sip-interface** configuration elements.

1. Access the **sip-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

2. **atcf-stn-sr**—Enter the value of the Session Transfer Interface, Single Radio (STN-SR). Your entry will resemble this example: tel:+1-237-555-9999. This value will be included in the g.3gpp.atcf media feature tag that the ATCF allocates in the REGISTER message.
3. **atcf-psi-dn**—Enter the value to use for the Public Service Identity Domain Name (PSI-DN). Your entry will resemble this example: sip:atcf.visited2.com. If configured, this value will be included in the g.3gpp.atcf media feature tag that the ATCF allocates in the REGISTER message. If you leave this parameter blank, the Oracle USM will set this value to the SIP interface accress.
4. **atcf-route-to-sccas**—If you leave this parameter set to disabled (default), the handover update, an INVITE, is routed to the IMS Core. If you set this parameter to enabled, the Oracle USM will route the handover update directly to the Service Centralization and Continuity Application Server (SCC-AS).
5. Type **done** to save your configuration.

Continue to and select the existing sip-interface configuration element targeted for this ATCF configuration:

```
ORACLE(sip-config)# exit
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)# select
<RealmID>:
1: public

selection: 1
ORACLE(sip-interface)#
```

6. **sip-atcf-feature**—Change this parameter from **disabled** (default) to **enabled** to turn on ATCF functionality for the ingress SIP interface.
7. Type **done** to save your configuration.

Emergency Access Transfer Function

The Emergency Access Transfer Function (EATF) is a logical, functional service defined in 3GPP TS 23.167, *IP Multimedia Subsystem (IMS) Emergency Sessions*, and TS 23.237, *IP Multimedia Subsystem (IMS) Service Continuity*; Stage 2. The EATF, essentially a special-purpose B2BUA, anchors emergency calls to enable access transfer between packet-switched and circuit-switched networks during eSR-VCC procedures when the LTE equipment is moving outside LTE coverage to either a 2G or 3G carrier network. Similar to the Access Transfer Control Function (ATCF) and ATGW (Access Transfer Gateway), the EATF is always located in the visited network when the user equipment is roaming.

Lacking this capability, the LTE equipment would be forced to re-establish the emergency session in the circuit-switched network through the legacy accesses (2G or 3G).

When the LTE equipment initiates a packet-switched emergency session, the INVITE is sent to the EATF thru the P-CSCF/E-CSCF (collocated on the SBC). This original INVITE is identified as an emergency session by the A-SBC/P-CSCF because it contains either an emergency short number (112, 991, and so forth) or an emergency service URN such as urn:service:sos.fire.

In the event that handoff to a circuit-switched network is required, the Mobile Switching Center (MSC) server initiates the transfer with a SIP INVITE containing an E-STN-SR (Emergency Session Transfer Number for Single Radio VCC) in the Request URI of the INVITE. Each network has a single E-STN-SR, essentially the telephone number of the EATF service, that is used exclusively for emergency session transfer access. The MSC directs the INVITE to the I-CSCF, which, in turn, which forwards the request directly to the EATF.

The EATF checks the E-STN-SR to determine that handoff to the circuit-switched network is requested and proceeds with the access transfer of the active session. The EATF associates the received SIP INVITE with an existing SIP session already anchored at the EATF using the instance-id feature tag. The EATF then sends a re-INVITE to the E-CSCF, which terminates the emergency session.

Once the session modification procedures are complete, as indicated by the reception of the SIP ACK request from the target access leg, the source access leg, previously established via IMS, is released.

Enabling EATF Capability

Use the following procedure to enable EATF operations.

1. Use the following procedure to move to sip-config configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

2. Use the **eatf-stn-sr** parameter to enable EATF operation and to provide E.164 telephone number of the EATF service.

   Provide the E.164 telephone number as either a tel: or a sip: URI. When confirming the E-STN-SR value, the match succeeds based solely on the E.164 portion regardless of the URI type.

   ```
   ACMEPACKET(sip-config)# eatf-stn-sr tel:+1-237-555-9999
   ACMEPACKET(sip-config)#
   ```

3. Use **done** and **exit** to complete configuration.

Monitoring EATF ATCF Sessions

The existing **show sipd srvcc** command has been enhanced to provide counts of successful and cancelled EATF sessions.

```
ORACLE# show sipd srvcc
SIP Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
ATCF success         -       -       1          1       1
ATCF failure         -       -       -          -       -
ATCF cancellation    -       -       -          -       -
EATF success         -       -       1          1       1
EATF cancellation
```

11

SIP Signaling Services

This chapter explains how to configure the Oracle USM to support Session Initiation Protocol (SIP) signaling services for hosted IP services applications. SIP is a text-based application-layer signaling protocol that creates, identifies, and terminates multimedia sessions between devices.

About the Oracle USM and SIP

The Oracle USM's support of SIP is broad in scope and varied in context. Beyond SIP interface configuration, the Oracle USM offers a multitude of configurations that control its utilization of the SIP protocol. Engineers involved in network design and administration need to identify which configurations are applicable and effective in their own environment and then apply those controls using the Oracle USM.

Mapping of Diversion Information Between Diversion and History-Info Headers

History-Info and Diversion are the two headers in SIP signaling used to convey information related to call transfer and call diversion. Although both provide call diversion information, they have different syntaxes, the main difference being that the chronology of events is reversed between the two headers. To date, Oracle USMs have provided mapping and interworking of the History-Info and Diversion headers through the use of an SPL plug-in. This feature implements this interworking functionality within the software by adding a new parameter to the **sip-interface** configuration element.

The History-Info header is the standard solution adopted by the Internet Engineering Task Force (IETF) for storing retargeting information. The non-standard Diversion header also is used in many existing network implementations. As both headers provide call forwarding needs but have different syntaxes, having both present in a signaling request can cause diverting information to be misinterpreted unless an interworking solution exists. The solution currently is the use of a Session Plug-in Language (SPL) plug-in; however, this feature embeds the functionality within the software by adding the new parameter **diversion-info-mapping-mode** to the **sip-interface** configuration element. This functionality enables the SBC to construct a new History-Info header for each Diversion header or a new Diversion header for each History-Info header, initialized with the value of the History-Info header, when the SBC receives a SIP INVITE which does not contain a to-tag in the To header.

Diversion and History-Info Headers Interworking Configuration

You can configure Oracle USMs to map call transfer and call diversion information between Diversion and History-Info headers.

To configure interworking between the Diversion and History-Info headers:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **diversion-info-mapping-mode**— Configure this parameter to specify how the Diversion and History-Info headers map to and work with each other on the interface. The default value is **none**.
   - **div2hist** — any Diversion headers in the initial INVITEs going out of this sip-interface will be converted to History-Info headers before sending
   - **force** — behavior is the same as **div2hist** when a Diversion header is present in the incoming INVITE. If there are no Diversion headers, a History-Info header for the current URI is added in the outgoing INVITE.
   - **hist2div** — any History-Info headers in the initial INVITEs going out of this sip-interface will be converted to Diversion headers before sending
   - **none** — no conversion applied (default)
4. Type **done** to save your configuration.

Recurse 305 Only Redirect Action

The Oracle USM has a SIP feature called redirect action. This is a feature that allows the Oracle USM, acting as a SIP Proxy or a Session Agent, to redirect SIP messages after receiving a SIP redirect (3xx) response. By default, the redirect-action parameter on the Oracle USM's sip-interface and session-agent on the is set to **recurse-305-only**.

Redirect Action Process

When the redirect-action parameter is set to proxy, the Oracle USM sends SIP Redirect responses back to the previous hop (back to the User Agent Client (UAC)) when the User Agent Server (UAS) is not a session agent. The URI in the Contact of the response is changed from the URI that was in the original request.

> **Note:**
>
> If the target of the request is a session agent, the session agent's redirect action supercedes that of the SIP interface.

When the redirect-action parameter is set to recurse, if the Oracle USM receives a SIP redirect (3xx) response on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 3xx response. The responses contain the same Contact URI that was in the original request sent to the UAS.

For example, if UAC X sends an INVITE to the Oracle USM set up as a SIP proxy, the Oracle USM forwards the INVITE to UAS Y (Y is not a session agent). Y then responds to the Oracle USM with a 3xx response (redirection message) with the same URI that was in the original request. This indicates to the Oracle USM that if it receives any future requests directed toward Y, that it should automatically redirect the request directly to Y. The Oracle USM then recurses, or repeatedly sends subsequent incoming messages to the Contact URI specified in the Header of the 3xx responses.

When the redirect-action parameter is set to recurse-305-only, if the Oracle USM receives a 305 SIP redirect response (Use Proxy) on the SIP interface, it automatically redirects all requests to the Contact URI specified in the 305 response. All other 3xx responses are sent back to the previous hop.

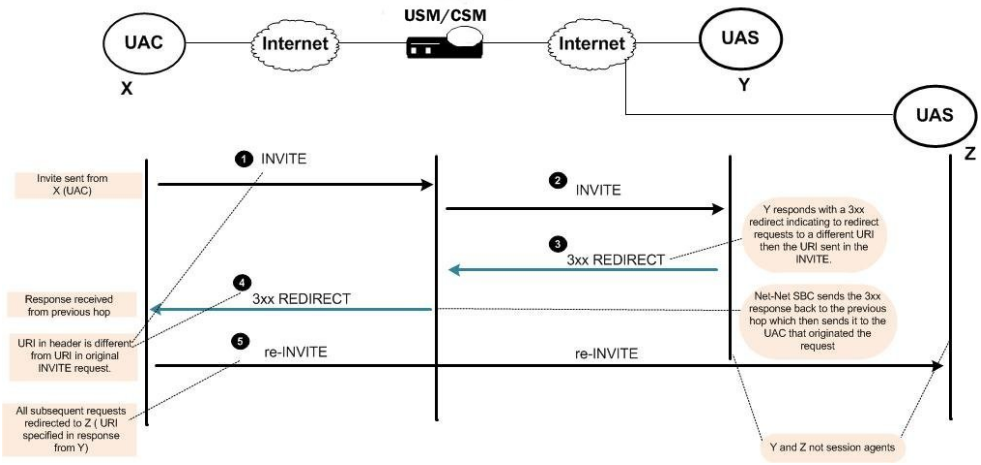
When the UAS is a session agent, the Oracle USM can send the SIP redirect response back to the UAC using the value in the session agent's redirect action field. If there are too many UASs to define as individual session agents, or if the UASs are Hosted NAT Traversal (HNT) endpoints, and SIP redirect responses need to be proxied for UASs that are not session agents, you can set the behavior at the SIP interface level.

Redirect-Action Set to Proxy

The following occurs if you set the **redirect-action** parameter to proxy on the Oracle USM:

1. X (UAC) sends an INVITE to the Oracle USM.
2. The Oracle USM forwards the INVITE to Y (UAS).
3. Y sends the 3xx REDIRECT response to the Oracle USM with a different URI in the message header.
4. The Oracle USM forwards the 3xx REDIRECT response to the previous hop. X receives the 3xx REDIRECT response from the previous hop.
5. X redirects all subsequent requests to the URI in the message header received from Y.

   The following illustration shows an example of a dialog between X, Y, Z, and the Oracle USM during a redirect-action session set to proxy.

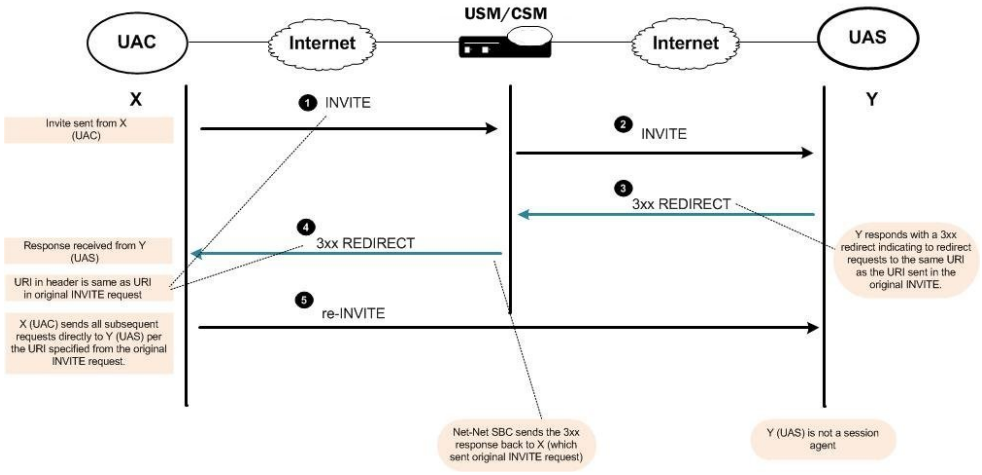


Redirect-Action Set to Recurse

The following occurs if you set the **redirect-action** parameter to recurse on the Oracle USM:

1. X (UAC) sends an INVITE to the Oracle USM.
2. The Oracle USM forwards the INVITE to Y (UAS).
3. Y sends the 3xx REDIRECT response to the Oracle USM with the same URI as the URI sent in the original request.
4. The Oracle USM forwards the 3xx REDIRECT response to X (UAC).
5. X (UAC) sends all subsequent requests directly to Y (UAS) per the URI specified from the original INVITE request.

   The following illustration shows an example of a dialog between X, Y, and the Oracle USM during a redirect-action session set to recurse.

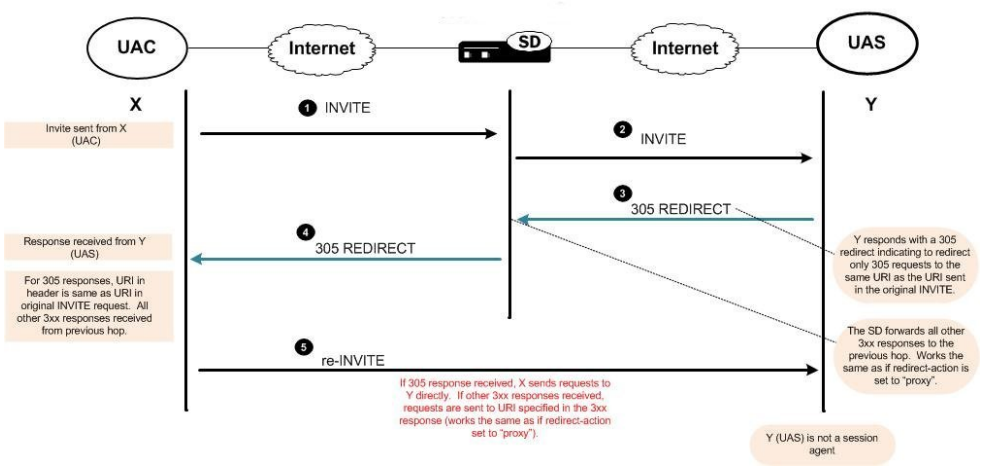


Redirect-Action Set to Recurse-305-Only

The following occurs if you set the **redirect-action** parameter to **recurse-305-only** on the Oracle USM:

1. X (UAC) sends an INVITE to the Oracle USM.
2. The Oracle USM forwards the INVITE to Y (UAS).
3. Y sends a 305 REDIRECT response to the Oracle USM with the same URI as the URI sent in the original request.
4. The Oracle USM forwards the 305 REDIRECT response to X (UAC).
5. If 305 response received, X sends requests to Y directly. If other 3xx responses received, requests are sent to URI specified in the 3xx response (works the same as if redirect-action set to proxy).

   The following illustration shows an example of a dialog between X, Y, and the Oracle USM during a redirect-action session set to recurse-305-only.



Embedded Routes in Redirect Responses

When the Oracle USM recurses as the result of a redirect (3xx) response, the server might need to specify one or more intermediate hops. These hops are reflected in the Contact header for the 3xx response using embedded route headers and look like this:

```
Contact: <sip:touser@server.example.com?Route=%3Cproxy.example.com%Blr%3E>
```

The Contact header shows that the request should be sent to server.example.com using proxy.example.com.

You can configure your Oracle USM to specify that embedded headers in 3xx Contact headers are to be included in new requests such that they are tied to a session agent representing the new target (server.example.com). This behavior requires you to set the **request-uri-headers** parameter.

However, you can also use the **use-redirect-route** in global SIP configuration's options parameter so that the embedded Route header is used as the next hop to receive the new request.

When you configure this new option, the Oracle USM constructs a new request using the redirect Contact, and the SIP URI from the Contact becomes the Request-URI. Then, the system inserts the embedded routes as Route headers in the, using the same order in which they appeared in the redirect Contact. Afterward, the Oracle USM determines the next hop in the same way it does with any other request. If the first route is a loose route (i.e., it has the lr URI parameter), then the Oracle USM sends a request to host indicated in the first route. Otherwise, strict routing applies, and the Oracle USM sends the request to the host indicated in the Request-URI.

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name.

   ```
   ORACLE(sip-config)# options use-redirect-route
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

SIP PRACK Interworking

When you configure your Oracle USM with PRACK interworking for SIP, you enable it to interwork between endpoints that support RFC 3262, *Reliability of Provisional Responses in the Session Initiation Protocol*, and those that do not.

As its title indicates, RFC 3262 defines a reliable provisional response extension for SIP INVITEs, which is the 100rel extension tag. While some endpoints do not support the RFC, other SIP implementations require compliance with it. A session setup between two such endpoints fails. However, you can configure your Oracle USM to supply the provisional response on behalf of endpoints that do not support it—and thereby enable sessions between those endpoints and the ones requiring RFC 3262 compliance.

You need to configure PRACK interworking for a SIP interface associated with the endpoints that need RFC 3262 support. To enabled the feature, you set the **100rel-interworking** option. The Oracle USM applies PRACK interworking for either the UAC or the UAS. TheOracle USM checks to see whether or not it needs to apply PRACK interworking when an INVITE arrives at the ingress or egress SIP interface with the option enabled. First, it checks the Require header for the 100rel tag; if not found there, it checks the Supported header.

Since there is a slight difference in the application of this feature between the UAC and UAS, this section explains both.

> **Note:**
>
> If SDP is included in a PRACK request sent to a SIP interface where PRACK interworking is enabled, it will not be responded to, nor will any SDP be included in the locally-generated 200 OK to that PRACK.

UAC-Side PRACK Interworking

The Oracle USM applies PRACK interworking on the UAC side when:

- A SIP INVITE does not contain a 100rel tag in a Require or Supported header
- The ingress SIP interface is enabled with the **100rel-interworking** option
- The UAS fails to send reliable provisional responses

When it is to forward a non-reliable response to a UAC that requires RFC 3262 support, the Oracle USM converts the non-reliable response to a reliable one by adding the 100rel tag to the Require header and adding an Rseq header to the response. Further, the Oracle USM adds a

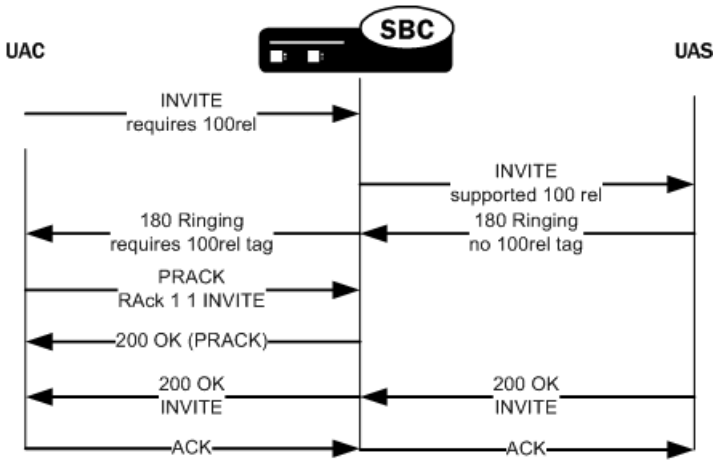
Require header (complete with the100rel tag) if there is not one already in the response, and then also adds Rseq header.

Note that the Oracle USM sets the value of the Rseq header as 1 for the first provisional response, and then increments it by 1 for each subsequent provisional response. It also adds the PRACK method to the Allow header when that header appears.

The Oracle USM retransmits the converted reliable provisional response in accordance with RFC 3262, until it receives a PRACK request. For the initial timeout for retransmission, the Oracle USM uses the value you set in the **init-timer** parameter in the global SIP configuration. It stops retransmitting when either it receives a transmission, or when the ingress SIP interface's trans-expire timer elapses.

If it never receives a PRACK, the Oracle USM does not generate an error response to the INVITE, relying instead on the downstream UAS to produce a final response.

The call flow for this application looks like this:



UAS-Side PRACK Interworking

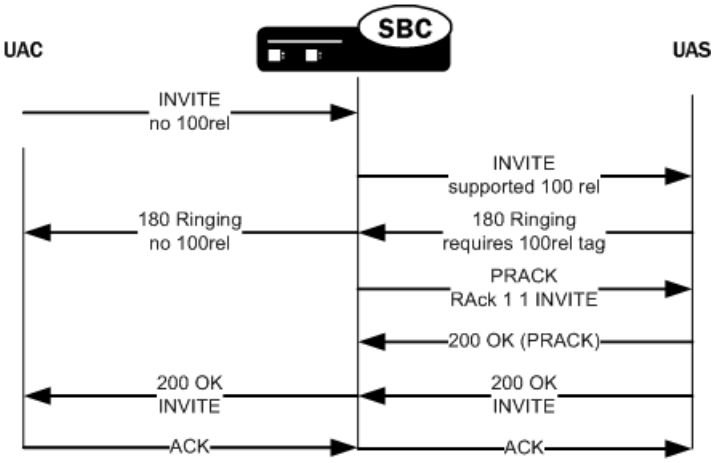
The Oracle USM applies PRACK interworking on the UAS side when:

- A SIP INVITE contains the 100rel tag in a Require or Supported header
- The egress SIP interface is enabled with the **100rel-interworking** option
- The UAS does send reliable provisional responses

When the UAC does not support RFC 3262, the Oracle USM generates a PRACK request to acknowledge the response. It also converts the response to non-reliable by removing the 100 rel tag from the Require header and removing the RSeq header from the response.

In the case of the UAS, the Oracle USM matches the PRACK to a converted reliable provisional response using the PRACK's RAck header. If it finds a matching response, the Oracle USM generates a 200 OK to the PRACK. And if it finds no match, then it generates a 481 Call Leg/Transaction Does Not Exist response. The Oracle USM generates a 400 Bad Request response if either the RAck is not in the PRACK request or it is not formatted properly.

The call flow for this application looks like this:



PRACK Interworking Configuration

You enable PRACK interworking for ingress and egress SIP interfaces. Be sure you know on what side, ingress or egress, you need this feature applied.

To configure PRACK interworking for a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. If you are editing an existing configuration, select the one on which you want to enable this feature.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **100rel-interworking** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +100rel-interworking
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

Global SIP Timers

This section explains how to configure SIP retransmission and expiration timers.

> **Note:**
>
> you can also set timers and counters per SIP interface.

Overview

SIP timers define the transaction expiration timers, retransmission intervals when UDP is used as a transport, and the lifetime of dynamic TCP connections. The retransmission and expiration timers correspond to the timers defined in RFC 3261.

- init timer: is the initial request retransmission interval. It corresponds to Timer T1 in RFC 3261.
  This timer is used when sending requests over UDP. If the response is not received within this interval, the request is retransmitted. The retransmission interval is doubled after each retransmission.
- max timer: is the maximum retransmission interval for non-INVITE requests. It corresponds to Timer T2 in RFC 3261.
  The retransmission interval is doubled after each retransmission. If the resulting retransmission interval exceeds the max timer, it is set to the max timer value.
- trans expire: is the transaction expiration timer. This value is used for timers B, D, F, H and J as defined in RFC 3261.
- invite expire: defines the transaction expiration time for an INVITE transaction after a provisional response has been received. This corresponds to timer C in RFC 3261.
  If a final response is not received within this time, the INVITE is cancelled. In accordance with RFC 3261, the timer is reset to the invite expire value when any additional provisional responses are received.
- Inactive dynamic conn timer defines the idle time of a dynamic TCP connection before the connection is torn down. Idle is defined as not transporting any traffic. There is no timer in RFC 3261 corresponding to this function.

Timers Configuration

To configure timers:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **init-timer**—Enter the initial timeout value in milliseconds for a response to an INVITE request, and it applies to any SIP request in UDP. In RFC 3261, this value is also referred to as TIMER_T1. The default is **500**. The valid range is:

   - Minimum—0

- Maximum—999999999

5. **max-timer**—Enter the maximum transmission timeout (T2) for SIP in milliseconds.

   When sending SIP over UDP, a re-transmission timer is used. If the timer expires and the message is re-transmitted, the re-transmission timer is then set to twice the previous value (but will not exceed the maximum timer value). Using the default values of 500 milliseconds and 4000 milliseconds, the re-transmission timer is 0.5, then 1, 2, and finally 4. The incrementing continues until the transmission expire timer activates. The default is **4000**. The valid range is:

   - Minimum—0
   - Maximum—999999999

6. **trans-expire**—Enter the transaction expire timeout value (Timer B) in seconds to set the time for SIP transactions to live. The same value is used for Timers D, F, H and J. The default is **32**. The valid range is:

   - Minimum—0
   - Maximum—999999999

7. **invite-expire**—Enter the invite expire timeout value (Timer C) in seconds to indicate the time for SIP client transaction will live after receiving a provisional response. The default is **180**. The valid range is:

   - Minimum—0
   - Maximum—999999999

8. **inactive-dynamic-conn**—Enter the inactive dynamic connection value in seconds to set the time limit for inactive dynamic connections.

   If the connection between the SIP proxy and a session agent is dynamic (for example, through dTCP), and the connection has been idle for the amount of time specified here, the SIP proxy breaks the connection. Idle is defined as not transporting any traffic. The default value is **32**. The valid range is:

   - Minimum—0
   - Maximum—999999999

   > **Note:**
   >
   > Setting this parameter to 0 disables this parameter.

   The following example shows SIP config timer values for a peering network. Some parameters are omitted for brevity.

```
sip-config
        state                         enabled
        operation-mode                dialog
dialog-transparency        disabled
home-realm-id                  acme
        egress-realm-id
        nat-mode                      Public
        registrar-domain
        registrar-host
        registrar-port                0
        init-timer                    500
        max-timer                     4000
```

```
trans-expire                32
invite-expire               180
inactive-dynamic-conn       32
```

SIP Timers Discreet Configuration

Previous releases controlled various SIP timers with a single ACLI command, **trans-expire**, available in both sip-config and sip-interface modes. When executed in sip-config mode, the command essentially established a global default transaction expiration timer value. Executed at the sip-interface level, the command established a local, interface-specific value that overrode the global default.

Specific timers controlled by **trans-expire** are as follows:

Timer B, the INVITE transaction timeout timer, defined in Section 17.1.1.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer D, the Wait-Time for response retransmitals timer, defined in Section 17.1.1.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer F, the non-INVITE transaction timeout timer, defined in Section 17.1.2.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer H, the Wait-Time for ACK receipt timer, defined in Section 17.2.1 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

Timer J, the Wait-Time for non-INVITE requests timer, defined in Section 17.2.2 and Appendix A of RFC 3261, *SIP: Session Initiation Protocol*.

A new ACLI command (**initial-inv-trans-expire**) that enables user control over SIP Timer B for initial INVITE transactions. Other timers, namely B for non-initial INVITEs, D, F, H, and J remain under the control of **trans-expire**.

Use **initial-inv-trans-expire** in the sip-config configuration mode, to establish a global, default transaction timeout value (expressed in seconds) used exclusively for initial INVITE transactions.

```
ORACLE(sip-config)# initial-inv-trans-expire 4
ORACLE(sip-config)#
```

Allowable values are integers within the range 0 (the default) through 999999999. The default value, 0, indicates that a dedicated INVITE Timer B is not enabled. Non-default integer values enable a dedicated Timer B and set the timer value.

The default value retains compatibility with previous operational behavior in that Timers B, D, F, H, and J all remain subject to the single timer value set by **trans-expire**. However, when **initial-inv-trans-expire** is et to a supported non-zero value, SIP Timer B as it applies to initial INVITEs, assumes that value rather than the value assigned by **trans-expire**. This functionality is available in both sip-config and in sip-interface objects.

If a dedicated Timer B is enabled at the sip-config level, you can use **initial-inv-trans-expire** in the sip-interface configuration mode, to establish a local interface-specific Timer B timeout value that overrides the global default value.

```
ORACLE(sip-interface)# initial-inv-trans-expire 8
ORACLE(sip-interface)#
```

Session Timer Support

The Oracle USM partially supports RFC4028 by establishing session timers without participating in the session timer negotiation.

When a 2xx response to a Session Refresh Request arrives, the Oracle USM will start a new timer or refresh the existing timer using the value of the Session-Expires header. When the session timer expires, the Oracle USM will send a BYE to both the upstream and downstream endpoints.

When accounting is configured, the Oracle USM will also send a RADIUS STOP record with Acct-Terminate-Cause=Session-Timeout.

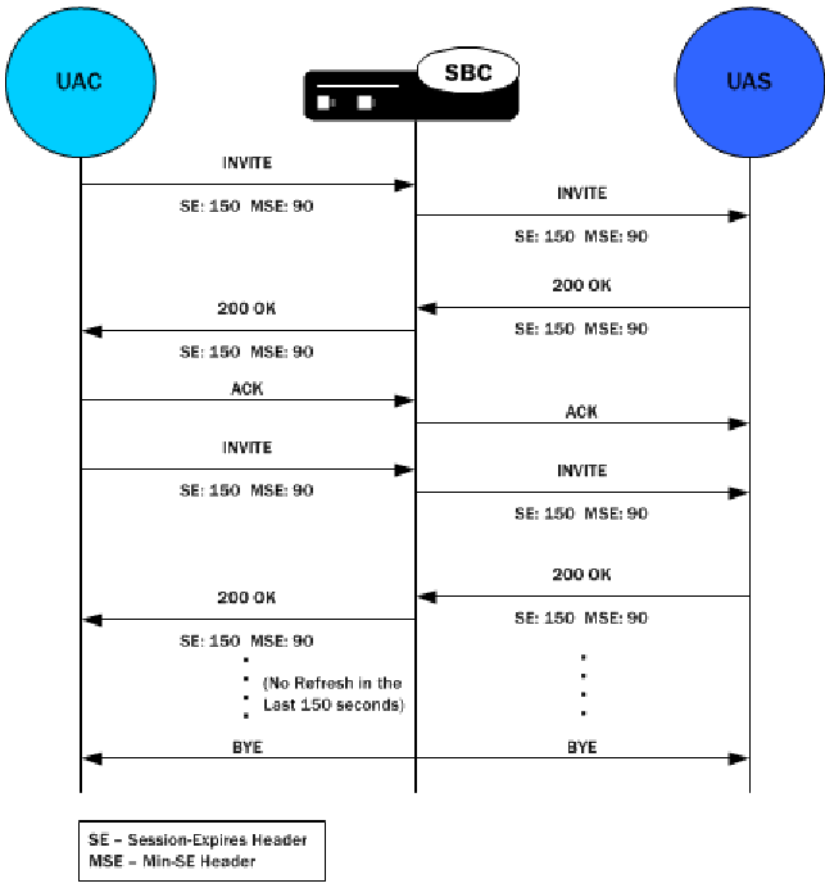
Call Flow Example

The UAS obtains the value from the Session-Expires header field in a 2xx response to a session refresh request that it sends.

Proxies and UACs obtain this value from the Session-Expires header field in a 2xx response to a session refresh request that they receive.

Once the session timer runs out, the Oracle USM sends a BYE to both the UAC and the UAS to clear the session.

Enable this feature by adding the **session-timer-support** option to the sip config.



SIP Per-User CAC

The Oracle USM's call admission control (CAC) supports an enhanced degree of granularity for SIP sessions.

Without this feature enabled, the Oracle USM performs call admission control (CAC) based on:

- Bandwidth limits configured in realms and nested realms
- Number of media flows available through the steering pool per realm
- Number of inbound sessions configured for a SIP session agent
- Number of total sessions (inbound and outbound) per SIP session agent
- Use of the Oracle USM's support for common open policy service (COPS), allowing the Oracle USM to perform CAC based on the policies hosted in an external policy server

These methods provide a basic level of call admission control in order to ensure that a SIP session agent's capacity is not exceeded. You can also ensure that signaling and media bandwidth capacities are not exceeded for physical trunks and peers.

With this feature enabled, the Oracle USM changes behavior so that it will only allow the configured number of calls or total bandwidth to and from each user in a particular realm. The overall realm bandwidth and steering pool limits still apply, and as before, the Oracle USM still rejects users who might be within their CAC limitations if accepting them with exceed the bandwidth limitations for parent or child realms and steering pools.

For SIP sessions, the Oracle USM now keeps track of the amount of bandwidth a user consumes and the number of active sessions per address of record (AoR) or per IP address, depending on the CAC mode you select (either aor or ip). When an endpoint registers with the Oracle USM, theOracle USM allots it a total amount of bandwidth and total number of sessions.

This section describes the details of how SIP per user CAC works.

You should note that the functionality this section describes only works if you enable registration caching on your Oracle USM.

For SIP sessions, the Oracle USM now keeps track of the amount of bandwidth a user consumes and the number of active sessions per address of record (AoR) or per IP address, depending on the CAC mode you select (either aor or ip). When an endpoint registers with the Oracle USM, the Oracle USM allots it a total amount of bandwidth and total number of sessions.

Per User CAC Modes

There are three modes that you can set for this feature, and each has an impact on how the other two per-user-CAC parameters are implemented:

- none—No per user CAC is performed for users in the realm.
- aor—The Oracle USM performs per user CAC according to the AoR and the contact associated with that AoR for users in the realm.
- ip—The Oracle USM performs per user CAC according to the IP address and all endpoints that are sending REGISTER messages from the IP address for users in the realm.

Per User CAC Sessions

You can set the number of CAC for sessions per user in the realm configuration. Depending on the CAC mode you set, the sessions are shared between contacts for the same AoR or the endpoints behind the same IP address.

When it receives an INVITE, the Oracle USM determines the registration entry for the calling endpoint and the registration for the called endpoint. It then decides if session can be established between the two. If it can, the Oracle USM establishes the session and changes the active session count for the calling and called endpoints. The count is returned to its original value once the session is terminated.

Per User CAC Bandwidth

You can set the per user CAC bandwidth in realm configuration, too, and it is handled much the same way that the sessions are handled. That is, depending on the CAC mode you set, the bandwidth is shared between contacts for the AoR or the endpoints behind the same IP address. All endpoints must be registered with the Oracle USM.

When it receives a Request with SDP, the Oracle USM checks to see if there is enough bandwidth for the calling endpoint and for the called endpoint. The Oracle USM assumes that the bandwidth usage is symmetric, and it uses the maximum bandwidth configured for the codec that it finds in the Request. In the event that there are multiple streams, the Oracle USM determines the total bandwidth required for all of the streams. If the required bandwidth exceeds what is available for either endpoint, the Oracle USM rejects the call (with a 503 error response). If the amount of available bandwidth is sufficient, then the used bandwidth value is increased for both the registered endpoints: calling and called. Any mid-session requests for changes in bandwidth, such as those caused by modifications in codec use, are handled the same way.

The Oracle USM also keeps track of the bandwidth usage on a global level. When the call terminates, the bandwidth it was consuming is returned to the pool of available bandwidth.

Notes on HA Nodes

This feature has been implemented so that a newly active system is able to perform SIP per user CAC. The standby Oracle USM is updated with the appropriate parameters as part of the SIP session update.

SIP per User CAC Configuration

Note that you must enable registration caching for this feature to work.

To configure SIP per user CAC:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. Select the realm where you want to want to add SIP per user CAC.

   ```
   ORACLE(realm-config)# select
   ```

5. **user-cac-mode**—Set this parameter to the per user CAC mode that you want yo use. The default value is **none**. The valid values are:
   - **none**—No user CAC for users in this realm
   - **aor**—User CAC per AOR
   - **ip**—User CAC per IP
6. **user-cac-sessions**—Enter the maximum number of sessions per user for dynamic flows to and from the user. The default is **0**. Leaving this parameter set to its means that there is unlimited sessions, meaning that the per user CAC feature is disabled in terms of the constraint on sessions. The valid range is:
7. Minimum—0
8. Maximum—999999999
9. **user-cac-bandwidth**—Enter the maximum bandwidth per user for dynamic flows to and from the user. The default is **0** and leaving this parameter set to the default means that there is unlimited bandwidth, meaning that the per user CAC feature is disabled in terms of the constraint on bandwidth. The valid range is:
   - Minimum—0
   - Maximum—999999999

SIP Per-Realm CAC

Building on the Oracle USM's pre-existing call admission control methods, CAC can be performed based on how many minutes are being used by SIP or H.323 calls per-realm for a calendar month.

In the realm configuration, you can now set a value representing the maximum number of minutes to use for SIP and H.323 session using that realm. Although the value you configure is in minutes, the Oracle USM performs CAC based on this value to the second. When you use this feature for configurations with nested realms, the parent realm will have the total minutes for all its child realms (i.e., at least the sum of minutes configured for the child realms).

The Oracle USM calculates the number of minutes used when a call completes, and counts both call legs for a call that uses the same realm for ingress and egress. The total time attributed to a call is the amount of time between connection (SIP 200 OK) and disconnect (SIP BYE), regardless of whether media is released or not; there is no pause for calls being placed on hold.

If the number of minutes is exhausted, the Oracle USM rejects calls with a SIP 503 Service Unavailable message (including additional information "monthly minutes exceeded). In the event that the limit is reached mid-call, the Oracle USM continues with the call that pushed the realm over its threshold but does not accept new calls. When the limit is exceeded, the Oracle USM issues an alarm and sends out a trap including the name of the realm; a trap is also sent when the alarm condition clears.

> **Note:**
>
> The Oracle USM does not reject GETS/NSEP calls based on monthly minutes CAC.

You can change the value for minutes-based CAC in a realm configuration at any time, though revising the value downward might cause limits to be reached. This value resets to zero (0) at the beginning of every month, and is checkpointed across both system in an HA node. Because this data changes so rapidly, however, the value will not persist across and HA node if both systems undergo simultaneous failure or reboot.

You can use the ACLI **show monthly minutes <realm-id>** command (where **<realm-id> is** the realm identifier of the specific realm for which you want data) to see how many minutes are configured for a realm, how many of those are still available, and how many calls have been rejected due to exceeding the limit.

SIP per Realm CAC Configuration

This section shows you how to configure minutes-based CAC for realms and how to display minutes-based CAC data for a specific realm.

Enabling Realm-Based CAC

Note that setting the new monthly-minutes parameters to zero (0), or leaving it set to its default of 0, disables this feature.

To configure minutes-based CAC:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to want to add SIP per user CAC.

   ```
   ORACLE(realm-config)# select
   ```

5. **monthly-minutes**—Enter the number of minutes allowed during a calendar month in this realm for SIP and H.323 calls. By default, this parameter is set to zero (0), which disabled monthly minutes-based CAC. You can enter a value as high as 71582788.

6. Save and activate your configuration.

Viewing Realm-Based CAC Data

Use the ACLI show monthly-minutes command to see the following information:

- How many minutes are configured for a realm

- How many of those are still available
- How many calls have been rejected due to exceeding the limit
  To view information about SIP per user CAC using the IP address mode:

1. In either User or Superuser mode, type **show monthly-minutes <realm-id>**, a Space, and the IP address for which you want to view data. Then press Enter. The **<realm-id>** is the realm identifier for. the realm identifier of the specific realm for which you want data

   ```
   ORACLE# show monthly-minutes private_realm
   ```

SIP Options Tag Handling

This section explains how to configure SIP options on a global or per-realm level and how to specify whether the feature treatment applies to traffic inbound to or outbound from a realm, or both.

SIP extensions that require specific behavior by UAs or proxies are identified by option tags. Option tags are unique identifiers used to designate new options (for example, extensions) in SIP. These option tags appear in the Require, Proxy-Require, and Supported headers of SIP messages.

Option tags are compatibility mechanisms for extensions and are used in header fields such as Require, Supported, Proxy-Require, and Unsupported in support of SIP.

The option tag itself is a string that is associated with a particular SIP option (i.e., an extension). It identifies this option to SIP endpoints.

Overview

The SIP specification (RFC 3261) requires that the Oracle USM B2BUA reject any request that contains a Require header with an option tag the Oracle USM does not support. However, many of these extensions operate transparently through the Oracle USM's B2BUA. You can configure how SIP defines the Oracle USMs B2BUA treatment of specific option tags.

Also, there might be certain extensions that an endpoint indicates support for by including the option tag in a Supported header. If you do not want a given extension used in your network, the you can configure SIP option tag handling to remove the undesired option tag from the Supported header. You can also specify how option tags in Proxy-Require headers are to be treated.

Configuration Overview

You configure the SIP feature element to define option tag names and their treatment by the Oracle USM when the option tag appears in a Supported header, a Require header, and a Proxy-Require header. If an option tag is encountered that is not configured as a SIP feature, the default treatments apply. You only need to configure option tag handling in the SIP feature element when non-default treatment is required.

You can specify whether a SIP feature should be applied to a specific realm or globally across realms. You can also specify the treatment for an option based on whether it appears in an inbound or outbound packet. Inbound packets are those that are coming from a realm to the Oracle USM and outbound packets are those which are going from the Oracle USM to the realm.

The following tables lists the SIP option tag parameters you need to configure.

Parameter	Description
name	SIP feature tag name
realm	Realm name with which the feature will be associated. To make the feature global, leave the field empty.
support mode inbound	Action for tag in Supported header in an inbound packet.
require mode inbound	Action for tag in Require header in an inbound packet
proxy require mode inbound	Action for tag in Proxy-Require header in an inbound packet
support mode outbound	Action for tag in Supported header in an outbound packet
require mode outbound	Action for tag in Require header in an outbound packet
proxy require mode outbound	Action for tag in Proxy-Require header in an outbound packet

SIP Option Tag Handling Configuration

To configure SIP option tag handling:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-feature** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-feature
   ORACLE(sip-feature)#
   ```

   From this point, you can configure SIP option tags parameters. To view all sip-feature parameters, enter a **?** at the system prompt.

4. **name**—Enter a name for the option tag that will appear in the Require, Supported, or Proxy-Require headers of inbound and outbound SIP messages.

   You must enter a unique value.

   > **Note:**
   >
   > Valid option tags are registered with the IANA Protocol Number Assignment Services under Session Initiation Protocol Parameters. Because option tags are not registered until the SIP extension is published as a RFC, there might be implementations based on Internet-Drafts or proprietary implementations that use unregistered option tags.

5. **realm**—Enter the name of the realm with which this option tag will be associated. If you want to apply it globally across realms, leave this parameter blank.

6. **support-mode-inbound**—Optional. Indicate the support mode to define how the option tag is treated when encountered in an inbound SIP message's Supported header. The default value is **pass**. Valid values are:

   - **pass**—Indicates the B2BUA should include the tag in the corresponding outgoing message.

- **strip**—Indicates the tag should not be included in the outgoing message. Use strip if you do not want the extension used.

7. **require-mode-inbound**—Optional. Indicate the require mode to define how the option tag is treated when it is encountered in an inbound SIP message's Require header. The default value is **reject**. The valid values are:
   - **pass**—Indicates the B2BUA should include the tag in the corresponding outgoing message.
   - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.
8. **require-mode-inbound**—Optional. Indicate the require proxy mode to define how the option tag is treated when encountered in an incoming SIP message's Proxy-Require header. The default is reject. The valid values are:
   - **pass**—Indicates the B2BUA should include the tag in the corresponding outgoing message.
   - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.
9. **support-mode-outbound**—Optional. Indicate the support mode to define how the option tag is treated when encountered in an outbound SIP message's Supported header. The default value is pass. Valid values are:
   - **pass**—Indicates the B2BUA should include the tag.
   - **strip**—Indicates the tag should not be included in the outgoing message. Use strip if you do not want the extension used.
10. **require-mode-outbound**—Optional. Indicate the require mode to define how the option tag is treated when it is encountered in an outbound SIP message's Require header. The default value is **reject**. Valid values are:
    - **pass**—Indicates the B2BUA should include the tag.
    - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.
11. **require-mode-outbound**—Optional. Indicate the require proxy mode to define how the option tag is treated when encountered in an outgoing SIP message's Proxy-Require header. The default value is **reject**. The valid values are:
    - **pass**—Indicates the B2BUA should include the tag.
    - **reject**—Indicates the B2BUA should reject the request with a 420 (Bad Extension) response. The option tag is included in an Unsupported header in the reject response.

    The following example shows SIP option tag handling configured for non-default treatment of option tags.

```
sip-feature
        name                            newfeature
        realm                           peer-1
        support-mode-inbound            Strip
        require-mode-inbound            Reject
        proxy-require-mode-inbound      Pass
        support-mode-outbound           Pass
        require-mode-outbound           Reject
        proxy-require-mode-outbound     Reject
        last-modified-date              2004-12-08 03:55:05
```

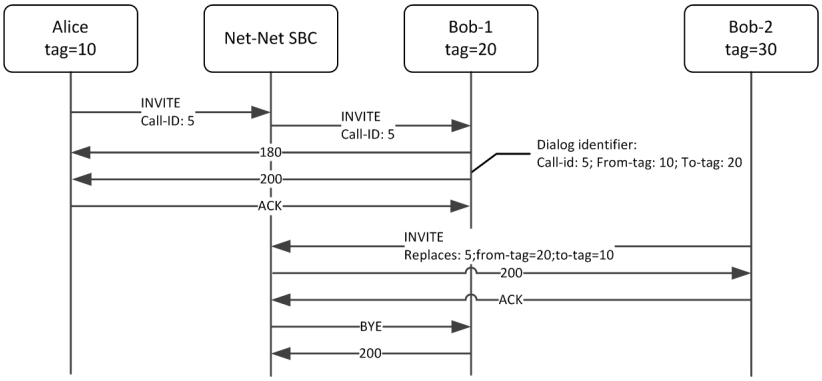
Replaces Header Support

The Oracle USM supports the Replaces: header in SIP messages according to RFC 3891. The header, included within SIP INVITE messages, provides a mechanism to replace an existing early or established dialog with a different dialog which can be used for services such as call parking, attended call transfer and various conferencing features.

The Replaces: header indicates the dialog it wishes to replace by containing the corresponding dialog identifier. The identifier includes the triplet of the from tag, to tag, and call id. The orientation of endpoint-created tags, as from-tag and to-tag will match each of the two dialogs for a standard call. Thus the Replaces: header from an endpoint that indicates it wants to assume the dialog between the Oracle USM and Bob-1 appears as:

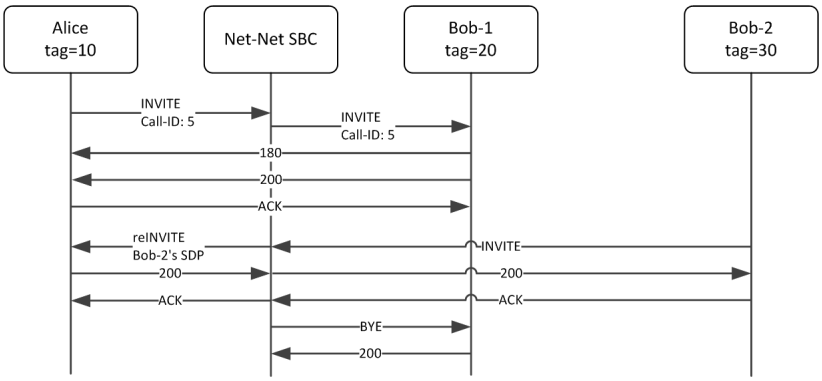
```
Replaces:5555;from-tag=20;to-tag=10
```

The Oracle USM validates that dialog identifier by matching an existing dialog and tries to install the new endpoint and remove the old endpoint by gracefully ending that dialog with a BYE. The replaces INVITE must come from an edpoint in the same realm as the endpoint it is replacing. If the UA sending the Replaces header is in a different realm as the original call leg (or indicates such architecture via a malformed Replaces: header), the Oracle USM replies to the Replaces: endpoint with a 481 Missing Dialog. Refer to the following diagram for the standard case.



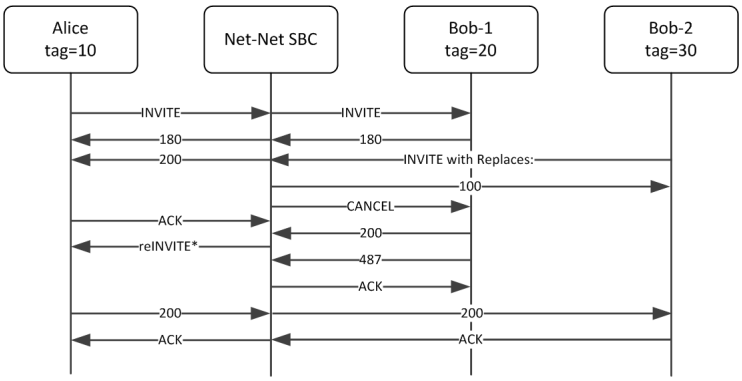
New SDP Parameters in INVITE with Replaces

When an INVITE with Replaces: header is received, the media parameters in the new SDP are compared against the SDP of the dialog to be replaced. If any portion of the SDPs (excluding the session-origin line) is different, then the media must be renegotiated. The Oracle USM sends a re-INVITE with the new SDP to the dialog opposite of the one being replaced as shown below. If the re-INVITE fails for any reason, then the original dialogs will remain.



Early Dialog Replacement

An INVITE with Replaces: header can replace an early dialog. That is, a dialog where the final 2xx class response to INVITE request has not arrived yet. The Oracle USM completes the originating side of the call with a 200 OK. The original dialog with the terminator is cancelled. SDP from the new terminator can be renegotiated if it changes.



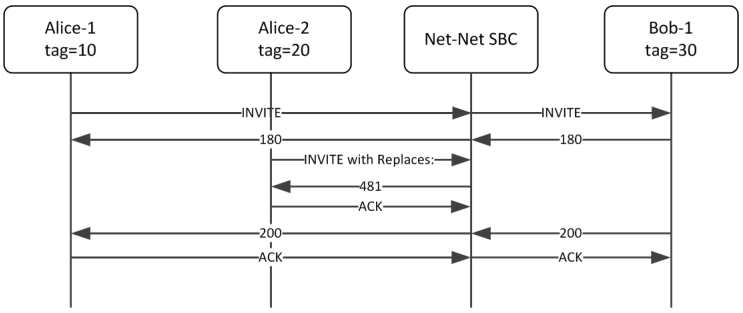
The SDP from the original 183 response is used for the 200 response back to the originator if present to complete the early transaction. If reliable provisional messages are used, then no SDP is included in the 200 response.

If no SDP is present in any of the provisional messages, then the Oracle USM constructs it from the original offer and modifying the IP port information for each c= and m= line with information from the INVITE with Replaces: header. If there are more m= lines in the original offer than ports from the INVITE with Replaces: header, then the extra ports are disabled with port value of 0. If no SDP was offered in the original INVITE, then the SDP from the INVITE with Replaces: header is used as the offer in the 200 OK.

If the SDP media parameters were compatible between the replaced and replacing SDPs, then media does not need to be renegotiated and no re-INVITE is created. If the re-INVITE fails, the original dialogs are torn down using a BYE for the original server dialog.

INVITE with Replaces in Early Dialog Server Side

The Oracle USM does not support replacing an early server dialog. It replies with a 481 (Dialog/Transaction does not exist) response to the endpoint requesting the replace.



Replace Header Configuration

Replaces: header support is configured in the session-agent, realm, or sip-interface via the sip-profile configuration element. sip-profiles are defined once and attached to a chosen interface, realm or session-agent.

The replace-dialogs parameter is set to either **enabled** or **disabled**. In addition, you may set this parameter to **inherit** which uses the next lower order of precedence object. If there are no sip-profiles referenced in the higher ordered object, or if all the replace-dialogs parameters are set to **inherit**, then the feature is disabled.

To configure Replaces: header support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter. **If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE (session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **replace-dialogs**—Set this parameter to **enabled** to enable Replaces: header support. A replaces parameter is also inservted in the Supported: header as sent into the realm where this sip profile is applied. You may also set this parameter to **inherit** for the element which this sip-profile is applied to, to inheret the value from the next-lower element.

5. Type **done** to save your work and continue.

Debugging

show sipd status

Includes Replaced Dialogs counts to show successfully replaced dialogs:

```
# show sipd status
17:41:38-142
SIP Status                -- Period -- -------- Lifetime --------
                  Active     High   Total      Total  PerMax    High
Replaced Dialogs     -        -       1          1       1
```

show sipd errors

Includes Replace Dialog Fails to show failed dialog replacements. This counter is incremented only when the dialog replacement attempt actually occurred but failed to successfully complete.

```
# show sipd errors
17:58:04-181
SIP Errors/Events             ---- Lifetime ----
                         Recent      Total  PerMax
Replace Dialog Fails          0          0       0
```

SIP Options

This section explains how you can configure a limited list of specialized SIP features and/or parameters called options. The options described here were developed to meet specific needs not addressed by the standard SIP configuration parameters. Not all users have a need for these options.

> **Note:**
>
> Oracle recommends checking with your Oracle representative before applying any of these options.

Overview

You can configure options for the SIP configuration and SIP interface. Both elements include a parameter (options) that you use to configure the options.

Global SIP Options

The following table lists the SIP options supported by the Oracle USM (USM).

Option	Description
add-error-to-tag=no	If present (even when set to no), suppresses the addition of an Acme tag on 3xx-6xx responses.
add-prov-to-tag=no	Prevents the USM from adding a tag parameter to the To header (to-tag) to non-100 provisional responses to INVITE requests. Used when a provisional (101-199) response is received from the UAS on a client transaction without a to-tag. By default, the USM adds the tag cookie in the response (as though it had a tag) sent back to the UAC for the associated server transaction. When you include this option in the SIP configuration, and the response from the UAS does not have a to-tag, the response forwarded to the UAC will not have a to-tag.
add-reg-expires	Causes an Expires header to always be included in a REGISTER response with the registration caching and HNT traversal functions of the USM. Use for endpoints that do not understand the Expires parameter in the Contact header.

Option	Description
add-ruri-user=<methods>	Causes a userinfo portion to be added to a Request-URI when one is not present. Used to support the OKI phone, which registers a Contact of just an IP-Address but rejects initial INVITEs if the Request_URI does not have a userinfo part. <methods> is a comma-separated list of methods to which the option should apply. If more than one method is listed, the list must be enclosed in quotes. This option only applies to out-of-dialog requests (no tag parameter in the To header). However, if ACK is listed, it will apply to all ACK requests because an ACK is always supposed to have a to-tag.
allow-notify-no-contact	Prevents the USM from rejecting NOTIFYs with a 400 Bad Request response.NOTIFY requests without Contact header are allowed to pass through the USM instead.
call-id-host=<host>	Causes the USM to include a host part (ID@host) in the Call-ID it generated. <host> is the hostname (or IP address) that is to appear in the host part of the Call-ID. If not specified, the SIP port address is used.
contact-endpoint=<param-name>	Defines a URL parameter to report the real Contact address of an endpoint in a REGISTER message forwarded to a registrar, when the USM is caching registration. (plain or HNT). If <param-name> is not specified, the default value endpoint is used. This parameter is added as a URL parameter in the Contact on the REGISTER message. In order for the registration cache to work properly, the softswitch/registrar is expected to include the endpoint parameter in the Request-URI of a SIP request it forwards to the address-of-record.
contact-firewall=<param-name>	Defines a URL parameter to report the NAT between the USM and the real Contact address of an endpoint in a REGISTRAR message forwarded to a registrar when the USM is doing registration caching for NHT. If <param-name> is not specified, the default value firewall is used. This parameter will be added as a URL parameter in the Contact on the REGISTER message. In order for the registration cache to work properly, the softswitch/registrar is expected to include the endpoint parameter in the Request-URI of any SIP request it forwards for the address-of-record.
disable-privacy	Prevents the change of the P-Preferred-Identity to P-Asserted-Identity and lets the P-Preferred-Identity go through unchanged.
drain-sendonly	Causes the USM to examine the SDP attributes and change sendonly mode to sendrecv. This causes the endpoint receiving the SDP to send RTP, which is required for HNT traversal endpoints to work with media servers. The USM sets up the flow so that RTP coming from the endpoint are dropped to prevent the UA that sent the sendonly SDP from receiving packets. See the option video-sbc-session also.

Option	Description
encode-contact=<prefix>	Causes the USM to encode Contact addresses into the userinfo part of the URI. It applies only to Contact address that usually get the maddr parameter. Use when the USM needs requests sent to the URI in the Contact sent instead to the USM. The host part of the URI will have the USM's address. The <prefix> serves as a place between the original userinfo and the encoded address. If a <prefix> is specified, a default of +SD is used. Without this option, the USM adds a maddr parameter.
fix-to-header	For requests that have the USM address in both the Request-URI and the To-URI, it sets the hostport of the To-URI to a local policy's next hop target on out-of-dialog requests (no to-tag). This is the default IWF behavior, even without this option configured.
forward-reg-callid-change	Addresses the case when an endpoint reboots and performs a third party registration before its old registration expires. During this re-registration, the contact header is the same as it was pre-reregistration. As a consequence of the reboot, the SIP Call-ID changes. In this situation, the USM does not forward the REGISTER to the registrar, because it believes the endpoint is already registered, based on a previous registration from the same Contact: header URI. To remedy this problem, the USM now keeps track of the Call-ID in its registration cache. A new option in the SIP interface configuration element forces the USM to forward a REGISTER message to the registrar when the Call-ID header changes in a REGISTER message received from a reregistering UAC.
global-contact	Addresses interoperability in the Dialog and Presence event packages that are used in hosted PBX and IP Centrex offerings. This option enables persistent URIs in the Contact headers inserted into outgoing SIP messages. If this option is not used, URIs placed in the Contact header of outgoing messages are only valid within the context of the dialog to which the message is associated.
ignore-register-service-route-oos	Prohibits a Register message from using a service route if that service route is an out-of-service session agent.
load-limit=<cpu percentage>	Defines the CPU usage percentage at which the USM should start rejecting calls. Default value is 90%.
lp-sa-match=<match strategy>	Changes the ways local policies and session agents match; accounts for realm in matching process. Strategy choices are: all, realm, sub-realm, interface, and network.
max-register-forward=<value>	Defines a limit (as assigned in the value field) of REGISTERs to be forwarded to the registrar. During each second, the sipd counts how many REGISTERs have been sent to the registrar. It checks the threshold when it receives a REGISTER from the UA and determines that less than half the real registration lifetime is left. If the number of REGISTERs forwarded (new and updates) in the current second exceeds the configured threshold, it will respond to the UA from the cache.

Option	Description
max-register-refresh=<value>	Defines the desired limit of REGISTER refreshes from all the UAs. Each second of time, sipd counts the number of REGISTER/200-OK responses sent back. When the threshold is exceeded, it increments the expire time (based on NAT interval) by one second and resets the count. By default no threshold is applied. The recommended value is somewhat dependent on the USM hardware used, but 300 can be used as an initial value.
max-routes=<number of routes>	Restricts the number of routes through which the sipd will iterate from a local policy lookup. For example, setting this option to 1 causes the USM to only try the first, best, route. Setting this option to 0, or omitting it, lets the USM use all of the routes available to it (with the priority scheme for route matching). When you test a policy using the test-policy ACLI command, this option is not recognized and all options that match the criteria are displayed.
max-udp-length=<maximum length>	Setting this option to zero (0) forces sipd to send fragmented UDP packets. Using this option, you override the default value of the maximum UDP datagram size (1500 bytes; sipd requires the use of SIP/TCP at 1300 bytes). You can set the global SIP configuration's max-udp-length=x option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
media-release=<header-name>[;<header-param>]	Enables the multi-system media release feature that encodes IP address and port information for the media streams described by SDP. It lets another USM decode the data to restore the original SDP, which allows the media to flow directly between endpoints in the same network (that is serviced by multiple USMs). The media release information can appear in the following places: • SIP header P-Media-Release: <encoded-media-interface-information> • Header parameter on a SIP header Contact: <sip:1234@abc.com> ; acme-media=<encoded-media-interface-information> • SDP attribute in the message body a=acme-media: <encoded-media-interface-information> Option includes the following: • <header-name> is SIP header in which to put the information or the special value sdp, which indicates the information should be put into the SDP. • <header-param> is the header parameter name in which to put the information or in the case of the special header name value sdp, it is the SDP attribute name in which to put the information. They identify to where the encoded information is passed. If you do not specify a header, P-Media-Release is used.

Option	Description
no-contact-endpoint-port	Enables the USM to add a URL parameter (defined as an argument to the contact-endpoint option) to the Contact headers of REGISTER messages that it forwards to the registrar when it performs registration caching. The value of the contact-endpoint URL parameter is the real address of the endpoint; and if the endpoint is behind a NAT, this includes the IP address and a port number. However, not all network entities can parse that port number, which is included unconditionally. This feature allows you to configure the exclusion of the port number. Despite the fact that you set this parameter in the global SIP configuration, it is applied only to SIP interfaces. However, you can set a contact-endpoint option in the realm configuration, on which this new parameter has no effect.
refer-to-uri-prefix=<prefix>	Defines a prefix to be matched against the userinfo part of Contact headers (config=), of which the USM should create a B2BUA map. This ensures that outgoing messages include the correct userinfo value. This option is used to enable add-on conferencing. 1. On the USM, set **refer-to-uri-prefix=<string>**, for example, **refer-to-uri-prefix="conf="**. 2. When the USM receives either an INVITE or 200 OK, it stores the Contact:sip:conf=<ID@IP:port> contained in the SIP message. 3. When the SBC receives a REFER in a separate call session, the config=<ID>@ IP2:port2 in this message is different. 4. The USM searches for the original conf=<ID> map, replaces the IP2:port2 with the stored IP:port, and forwards the updated REFER message.
reg-cache-mode=<mode>	Affects how the userinfo part of Contact address is constructed with registration caching. <mode> values are: • none: userinfo from the received (post NAT) Contact is retained • from: userinfo from the From header is copied to the userinfo of the forwarded Contact header • append: append the UA's Contact address into a cookie appended to the userinfo from the original Contact userinfo. For HNT, the NAT/firewall address is used. • append-from: takes userinfo from the From header and appends the encrypted address to the userinfo from the original Contact userinfo. For HNT, the NAT/firewall address is used. The from mode is used with softswitches that do not use the cookies used by the USM. It also helps limit the number of bytes in the userinfo; which might create duplicate contacts. For example, if the USM address is 1.2.3.4, both 1234@5.6.7.8 and 1234@4.3.2.1 will result in a USM contact of 1234@5.6.7.8.

Option	Description
reg-contact-user-random	Support the SIP random registered-contact feature. Gives the USM the ability to support endpoints that randomly change their contact usernames every time they re-register. Only applicable to operators who need to support the Japan TTC standard JJ-90.22 in specific applications. Applies to cases when an endpoint re-registers with a different contact username, but with the same hostname/IP address and the same address of record (AoR). Without this feature enabled, the USM forwards every re-registration to the registrar with the new contact information without it being considered a registration refresh. The USM forwards it to the Registrar using the same sd-contact as the previous registration. When you set this option, the USM does treat such a re-registration as a registration refresh when it is received prior to the half-life time for the specific contact. The USM also uses the new contact username for the Request-URI in requests it sends to the UA, and verifies that the UA uses the correct one when that USM is set to allow-anonymous registered mode. NOTE: The registration cache mode is set using the option reg-cache-mode, but regardless of how you configure it, the registration cache mode will be set to contact when SIP random registered-contact feature is enabled.
register-grace-timer	Makes the grace time for the SIP Registration configurable. You can configure the grace timer in seconds.
reinvite-trying=[yes]	Causes the USM to send a 100 Trying for re-INVITEs, which is normally suppressed. If you enter the option name but omit the value yes, the option is still active.
reject-interval=<value>	Acts as a multiplier to increase the value presented to the UAC in the Retry-After field. For example, if reject-interval=5 (reject interval is set to 10); at a 90% rejection rate the USM sends Retry-After: 45. When rejecting calls because of CPU load limiting, the USM adds a Retry-After parameter to the error response (typically 503 Service Unavailable). By default the USM sets the Retry-After value to be 1/10th of the current rejection rate.
reject-register=[no | refresh]	Allows REGISTER messages through even during load limiting. By default, REGISTER messages are subject to load limiting.
response-for-not-found=<response code>	Change the 404 Not Found generated by the USM to a different response code.
route-register-no-service-route	Controls how a UA is registered. Option can have three values: • route-register-no-service-route—This option prevents the use of the Service-Route procedure to route the Re-Register requests after the UA has initially registered. • route-register-no-service-route=all—Prevents the use of the Service-Route procedure to route the Re-Register requests for all messages, after the UA has initially registered. • route-register-no-service-route=refresh—Prevents the use of the Service-Route procedure to route the Re-Register requests for all refresh-register messages, but not de-register messages, after the UA has initially registered. Addition idle argument ensures that, when enabled, the USM follows the previously defined rules for idle calls, where idle means not engaged in any INVITE-based sessions. Sample syntax: route-register-no-service-route=refresh;idle

Option	Description
sdp-insert-sendrecv	When a call is initiated, the SDP communicates between call offerer and call answerer to determine a route for the media. Devices can be configured to only send media (“a=sendonly”), to only receive media (“a=recvonly”), or to do both (“a=sendrecv”). Some devices, do not disclose this information. With this option configured, when either the offerer or answerer does not disclose its directional attribute, the USM automatically inserts a sendrecv direction attribute to the media session.
set-inv-exp-at-100-resp	Set Timer C when a 100 Trying response is received (instead of waiting until 1xx (> 100) is received). If the USM does not receive a 100 Trying response within Timer B, the call should be dropped because there is a problem communicating with the next hop.
strip-domain-suffix-route	Causes sipd to strip any Router headers from the inbound messages coming to the external address of a SIP NAT; if the message contains a FQDN that matches the configured domain suffix for that SIP NAT.
video-sbc-session	Use with drain-sendonly for conference floor support. When configured with drain-sendonly and when the USM receives an SDP, the USM proxies the m=control and its related a= and c= unchanged. Although media streams are allocated for this m line, an actual flow is not set up. SDP received with the following: m=video a=sendonly is sent out as the following: m=video a=sendonly a=X-SBC-Session
session-timer-support	This option enables the USM to start the session timer for session refreshes coming from the UAC. The USM determines whether or not a session is active based on session refreshes or responses. It terminates the session when no session refreshes occur within the session timer interval.
session-timer-support	Enables RFC4028 session timer support.
inmanip-before-validate	Enables SIP Header Pre-processing for HMR.
proccess-implicit-tel-URI	Correctly appends coodie in REGISTER message when user=phone does not exist.
offerless-bw-media	Reserves appropriate bandwidth for an INVITE with no SDP.

SIP Interface Options

The following table lists the SIP interface options supported by the Oracle USM.

Option	Description
100rel-interworking	Enables RFC 3262, *Reliability of Provisional Responses in the Session Initiation Protocol* support.

Option	Description
contact-endpoint=<endpoint name>	The Oracle USM inserts the endpoint IP address and port into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value endpoint is used.
contact-firewall=<firewall name>	The Oracle USM inserts the firewall IP address and port into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value firewall is used.
contact-vlan=<VLAN/realm name>	The Oracle USMinserts the realm and VLAN ID into the Contact headers as messages egress using that SIP interface. The inserted data is the same as the information received in the Request or Response being forwarded. If the endpoint name is not specified, the default value vlan is used.
dropResponse	The Oracle USM drops responses by specified status codes. The option value can contain one or more status codes separated by semicolons. Error ranges can also be entered. If any of the response codes matches then a response is not sent. If the dropResponse option is set in both the sip-interface and the session-agent elements, the session-agent setting takes precedence.
max-udp-length=<maximum length>	Sets the largest UDP packers that the Oracle USM will pass. Packets exceeding this length trigger the establishment of an outgoing TCP session to deliver the packet; this margin is defined in RFC 3261. The system default for the maximum UDP packet length is 1500. You can set the global SIP configuration's max-udp-length=x option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
response-for-not-found=<response code>	Change the 404 Not Found generated by the SBC to a different response code.
strip-route-headers	Causes the Oracle USM to disregard and strip all route headers for requests received on a SIP interface.
upd-fallback	When a request needs to be sent out on the SIP interface for which you have configured this option, the Oracle USM first tries to send it over TCP. If the SIP endpoint does not support TCP, however, then the Oracle USM falls back to UDP and tries the request again.
via-header-transparency	Enables the Oracle USM to insert its Via header on top of the top-most Via header received from user equipment (UE). It then forwards it on to the IP Multimedia Subsystem (IMS) core with the original Via header now located as the bottom-most Via header. The Oracle USM still replaces the Contact and other header addresses with its own, and does not pass on the core's Via headers in outbound requests.
use-redirect-route	Use Route parameter in Contact header as next-hop as received in a 3xx response.

Option	Description
reg-via-proxy	Enables your Oracle USM to support endpoints that register using an intervening proxy.
lmsd-interworking	Enables 3GPP2 LMSD Interworking.
suppress-reinvite	Enables reINVITE supression.

SIP Session Agent Options

The following table lists the SIP session agent options supported by the Oracle USM.

Option	Description
dropResponse	The Oracle USMdrops responses by specified status codes. The option value can contain one or more status codes separated by semicolons. Error ranges can also be entered. If any of the response codes matches then a response is not sent. If the dropResponse option is set in both the sip-interface and the session-agent elements, the session-agent setting takes precedence.
trans-timeouts=<value>	Defines the number of consecutive non-ping transaction timeouts that will cause a session agent to be out of service. When the session agent is configured, i.e. when the PING options are defined, the value is 10. If not defined, the default value is 5. A Value of 0 prevents the session agent from going out of service because of a non-ping transaction timeout.
via-origin=<parameter-name>	Causes a parameter to be included in the top Via header of requests sent to the session agent. The parameter indicates the source IP address of the corresponding request received by the Oracle USM. <parameter-name> defines the name of the parameter. If not specified, the default value origin is used.
refer-reinvite	Enables SIP REFER with Replaces.

SIP Realm Options

The following table lists the SIP session agent options supported by the Oracle USM.

Option	Description
number-normalization	Applies to the SIP To URI. (Currently the Oracle USM supports number normalization on From and To addresses for both inbound and outbound call legs.) Number normalization includes add, delete, and replace string functions that result in consistent number formats. Number normalization occurs on ingress traffic, prior to the generation of accounting records or local policy lookups. (also applies for H.323 to SIP calls.)
refer-reinvite	Enables SIP REFER with Replaces.

SIP Realm Options Configuration

To configure options:

Labels enclosed in <> indicate that a value for the option is to be substituted for the label. For example, <value>. In order to change a portion of an options field entry, you must re-type the entire field entry.

1. Navigate to the options parameter in the SIP configuration or SIP interface elements.
2. Enter the following:

   ```
   options Space <option name>=”<value>”
   ```

   For example, if you want to configure the refer-to-uri-prefix option (the add-on conferencing feature):

   Type options, followed by a Space.

   Type refer-to-uri-prefix, followed by an equal sign (=).

   Type the opening quotation mark (“) followed by conf, another equal sign and the closing quotation mark.

   Press Enter.

   For example:

   ```
   options refer-to-uri-prefix=conf=
   ```

   If the feature value itself is a comma-separated list, it must be enclosed in quotation marks.

Configuring Multiple Options

You can enter a list of options for this field:

1. Type **options** followed by a space.
2. Within quotation marks, enter the feature names and values of the parameters you need. Separate each one with a comma.
3. Close the quotation marks.
4. Press Enter.

   For example:

   ```
   ACMEPACKET(sip-config)# options “refer-to-uri-prefix=“conf=”,encode-
   contact=“+SD”,add-ruri-user=INVITE,ACK”
   ```

Adding an Entry

Enter the new entry with a preceding plus (+) sign. For example:

```
options +response-for-not-found
```

This format allows previously configured options field values to remain intact without requiring re-entry of the entire field value.

SIP Security

This section provides an overview of Oracle USM’s security capability. Oracle USM security is designed to provide security for VoIP and other multi-media services. It includes access control, DoS attack, and overload protection, which help secure service and protect the network infrastructure (including the Oracle USM). In addition, Oracle USM security lets legitimate users to still place calls during attack conditions, protecting the service itself.

Oracle USM security includes the Net-SAFE framework's numerous features and architecture designs. Net-SAFE is a requirements framework for the components required to provide protection for the Session Border Controller (SBC), the service provider's infrastructure equipment (proxies, gateways, call agents, application servers, and so on), and the service itself.

Denial of Service Protection

The Oracle USM Denial of Service (DoS) protection functionality protects softswitches and gateways with overload protection, dynamic and static access control, and trusted device classification and separation at Layers 3-5. The Oracle USM itself is protected from signaling and media overload, but more importantly the feature allows legitimate, trusted devices to continue receiving service even during an attack. DoS protection prevents the Oracle USM host processor from being overwhelmed by a targeted DoS attack from the following:

- IP packets from an untrusted source as defined by provisioned or dynamic ACLs
- IP packets for unsupported or disabled protocols
- Nonconforming/malformed (garbage) packets to signaling ports
- Volume-based attack (flood) of valid or invalid call requests, signaling messages, and so on.
- Overload of valid or invalid call requests from legitimate, trusted sources

Levels of DoS Protection

The multi-level Oracle USM Denial of Service protection consists of the following strategies:

- Fast path filtering/access control: involves access control for signaling packets destined for the Oracle USM host processor as well as media (RTP) packets. The SBC accomplishes media filtering using the existing dynamic pinhole firewall capabilities. Fast path filtering packets destined for the host processor require the configuration and management of a trusted list and a deny list for each Oracle USM realm (although the actual devices can be dynamically trusted or denied by the Oracle USM based on configuration). You do not have to provision every endpoint/device on the Oracle USM, but instead retain the default values.
- Host path protection: includes flow classification, host path policing and unique signaling flow policing. Fast path filtering alone cannot protect the Oracle USM host processor from being overwhelmed by a malicious attack from a trusted source. The host path and individual signaling flows must be policed to ensure that a volume-based attack will not overwhelm the Oracle USM's normal call processing; and subsequently not overwhelm systems beyond it. The Oracle USM must classify each source based on its ability to pass certain criteria that is signaling- and application-dependent. At first each source is considered untrusted with the possibility of being promoted to fully trusted. The Oracle USM maintains two host paths, one for each class of traffic (trusted and untrusted), with different policing characteristics to ensure that fully trusted traffic always gets precedence.
- Host-based malicious source detection and isolation – dynamic deny list. Malicious sources can be automatically detected in real-time and denied in the fast path to block them from reaching the host processor.

Configuration Overview

NAT table entries are used to filter out undesired IP addresses (deny list). After the packet from an endpoint is accepted through NAT filtering, policing is implemented in the Traffic Manager

based on the sender's IP address. NAT table entries are used to distinguish signaling packets coming in from different sources for policing purposes.

You can configure deny rules based on the following:

- ingress realm
- source IP address
- transport protocol (TCP/UDP)
- application protocol (SIP)

You can configure guaranteed minimum bandwidth for trusted and untrusted signaling paths.

You can configure signaling path policing parameters for individual source addresses. Policing parameters include:

- peak data rate in bits per second
- average data rate in bits per second
- maximum burst size

SIP Unauthorized Endpoint Call Routing

The Oracle USM (USM) can route new dialog-creating SIP INVITEs from unauthorized endpoints to a session agent or session agent group; then rejection can occur based on the allow-anonymous setting for the SIP port. This type of provisional acceptance and subsequent rejection applies only to INVITEs; the USM continues to reject all other requests, such as SUBSCRIBE.

You might enable this feature if you have a network in which unauthorized SIP endpoints continually try to register even if the Oracle USM has previously rejected them and never will accept them. For instance, the user account associated with the endpoint might have been removed or core registrars might be overloaded.

SIP Unauthorized Endpoint Call Routing Configuration

You enable the routing of unauthorized endpoints to session agents and session agent groups that will reject them in the SIP interface configuration.

To enable SIP unauthorized endpoint call routing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **route-unauthorized-calls**—Enter the name (or IP address) of the session agent or session agent group to which you want calls from unauthorized endpoints routed. This parameter is blank by default, meaning the SIP unauthorized call routing feature is disabled.

   Remember your settings in the **allow-anonymous** parameter in the SIP port configuration provide the basis for rejection.

5. Save and activate your configuration.

SIP NAT Function

This section explains how to configure the optional SIP NAT function. You can configure the SIP NAT function if you need to translate IP address and UDP/TCP port information. The SIP NAT function also prevents private IP addresses in SIP message URIs from traveling through an untrusted network.

Overview

TheOracle USM is an intermediary device that provides NAT functions between two or more realms. It translates IP addresses between untrusted and trusted networks using NAT. A trusted network is inside the NAT, and a untrusted network is outside the NAT. A NAT also lets a single IP address represent a group of computers.

For SIP, the SIP NAT function on the Oracle USM does the following:

- routes SIP packets between the Oracle USM's SIP proxy (B2BUA) and external networks (or realms), including the translation of IP address and UDP/TCP port information.
- prevents private IP addresses in SIP message URIs from traveling through the untrusted network. SIP NAT either translates the private address to one appropriate for an untrusted address or encrypts the private address into the URI.

Packets arriving on the external address (at port 5060) are forwarded to the Oracle USM's SIP proxy with the source address changed to the home address (at port 5060). When the Oracle USM's SIP proxy sends packets to the home address (at port 5060), they are forwarded to the external proxy address (and external proxy port), with the source address changed to the external address (at port 5060).

> **Note:**
>
> The SIP config's NAT mode parameter works in conjunction with the SIP NAT function configuration. It identifies the type of realm in which the SIP proxy is located (public or private) and affects whether IPvr addresses in SIP messages are encoded.

The translation of URIs in the actual SIP message occurs as messages are received and sent from the Oracle USM's SIP proxy. For the messages being sent to the external network, the contents of the SIP message are examined after the translation to determine if the destination needs to be changed from the external proxy address to an address and port indicated by the SIP message. This process takes place so the request is sent to where the Request-URI or the Route header indicates, or so the response is sent to where the Via indicates.

NAT Modes

The specific addresses used in translating URIs in the SIP message depend on whether the Oracle USM is performing NAT functions for a trusted or untrusted network. This condition is determined by the NAT mode value you enter when you configure the SIP config element. The NAT modes are:

- **untrusted**—The SIP proxy is associated with an address for an untrusted network (the address value you entered when you configured the SIP interface's SIP port parameter), and the home address in the SIP NAT is the address of the external realm/network. When the URI contains the external address, it is translated to the SIP NAT's home proxy address (or to the SIP port address if the home proxy address field is empty). When a URI contains the external proxy address, it is translated to the home address.
  If the URI contains any other private address (matching the realm's address prefix, identified in the SIP NAT's realm ID), it is encrypted and the address is replaced with the home address value. If the URI contains a user part, a suffix consisting of the user NAT tag and the encrypted address is appended to the user part. For example, with a user NAT tag value of -private-, the private URI of sip@123192.169.200.17:5060 will become the public URI of sip:123-private-eolmhet2chbl3@172.16.0.15.

  If there is no user part, the host consists of the host NAT tag followed by the encrypted address and the domain suffix. A maddr parameter equal to the home address (or received in the case of a Via header) is added to the URI. For example, with a host NAT tag value of PRIVATE- and a domain suffix value of private.com, the private URI of sip:192.168.200.17:5060 will become the public URI of sip:PRIVATE-eolmhet2chbl3.private.com:5060;maddr=172.16.0.15.
- **trusted**—The SIP proxy is on a trusted network (the address value you entered when you configured the SIP interface's SIP port parameter), and the SIP NAT's external address is the public address of the external realm/network. When the URI contains the home address value, it is translated to the value set for the external proxy address. When the URI contains the SIP proxy's address, it is translated to the external address. If the URI contains any other private address (matching the realm's address prefix, identified in the SIP NAT's realm ID), the private address is encrypted and the address is replaced with the external address.

> **Note:**
>
> Do not use the home proxy address value with private NAT functioning.

Adding a maddr Parameter to a URI

When you configure a SIP interface, you can configure the contact mode. The contact mode sets the contact header routing mode, which determines how the contact address from a trusted network is formatted. You set the contact mode to add a maddr parameter equal to the SIP proxy to the URI in the Contact header. For example, the URI from the prior example (sip:192.168.200.17:5060) becomes sip:123-trusted-eolmhet2chbl3@172.16.0.15;maddr=172.16.0.12.

> **Note:**
>
> For SIP elements that do not support the maddr parameter, configure a Contact mode as none.

You might require this encryption to cause other SIP elements in the untrusted network to send requests directly to the SIP proxy. Otherwise, the requests are sent to the home address. However, responses sent by the SIP proxy will have the SIP proxy's source address, rather than the home address. Some SIP elements might drop responses that come from a IP address different from the one to which the request is sent.

About Headers

You can specify which SIP headers you want effected by the SIP NAT function. The URIs in these headers are translated and encrypted, the encryption occurs according to the rules of this SIP NAT function.

You can enter header values by using either the full header name or its corresponding abbreviation, if applicable. The following table lists the available headers and their corresponding abbreviations

Header	Abbreviation
Call-ID	i
Contact	m
From	f
Record-Route	none
Route	none
Ready-To	none
Replaces	none
Refer-To	r
To	t
Via	v

SIP sessions are terminated and re-originated as new sessions as they are routed through the Oracle USM. Among the actions performed, SIP headers are modified to prevent the transmission of IP address and route information.

Replacing Headers

In the SIP signaling message, any Via headers are stripped out and a new one is constructed with the Oracle USM's IP address in the sent-by portion. If a Contact header is present, it is replaced with one that has the Oracle USM's IP address. All other headers are subject to NATing based on the following rules:

- The Request-URI is replaced with the next hop's IP or FQDN address.
- All other headers are replaced based on the two SIP NAT function SIP NAT function rules

Mapping FQDNs

The Oracle USM maps FQDNs that appear in the certain headers of incoming SIP messages to the IP address that the 12:00 inserts in outgoing SIP contact headers. The mapped FQDNs are restored in the SIP headers in messages that are sent back to the originator.

This feature is useful to carriers that use IP addresses in the SIP From address to create trunk groups in a softswitch for routing purposes. When the carrier's peer uses FQDNs, the carrier is forced to create trunk groups for each possible FQDN that it might receive from a given peer. Similarly, this can apply to SIP Contact and P-Asserted-Identity headers.

SIP NAT Function Cookies

Cookies are inserted to hide that information is coming from a realm external to the home realm. They are used when information needs to be placed into a given element of a SIP message that must also be seen in subsequent SIP messages within a flow. When forwarding a SIP message, the Oracle USM (USM) encodes various information in the outgoing message, which is passed from one side to another in SIP transactions.

SIP NAT function cookies let the USM hide headers, IPv4 addresses, and SIP URIs. These cookies are included when certain conditions are present in USM SIP transactions.

Oracle's SIP NAT function cookies can be used in the userinfo, host, URL parameter, and tel URL parameter portions of the SIP message.

userinfo

The Oracle USM places a cookie in the userinfo portion of a SIP URI when a SIP header contains a SIP URI, and includes that header type in the list of headers to be hidden (encrypted) in the associated SIP NAT function. The cookie for the userinfo portion is the following:

```
[user nat tag][encrypted 13-byte host IP][encrypted 13 byte maddr IP (if
present)]
```

where:

- [user nat tag] refers to the SIP NAT function's original user NAT tag field.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.

With a user NAT tag of -acme, the following SIP-URI:

```
sip:6175551212@192.168.1.100
```

might be translated into:

```
sip:6175551212-acme-pfi1s7n2pstna@172.16.1.10
```

> **Note:**
>
> Multiple additional cookies might be appended with each hop (for example, from the external proxy to the home proxy and back).

host

When hiding IP addresses in a SIP message, the SIP NAT function generates the following cookie for a SIP-URI with no userinfo portion:

```
[host nat tag][encrypted 13-byte host IP][encrypted 13 byte maddr IP (if
present)][domain suffix]
```

where:

- [host nat tag] refers to the SIP NAT function's host NAT tag.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.
- [domain suffix] refers to the SIP NAT function's domain suffix field.

With a SIP NAT function's host tag of ACME- and a domain suffix of .acme.com, the following SIP header:

Via: SIP/2.0/UDP 192.168.1.100:5060

might be translated into the following:

```
Via: SIP/2.0/UDP ACME-pfi1s7n2pstna.acme.com
```

URL Parameter

If the SIP NAT function's use url parameter field has a value of from-to or all, the SIP NAT function places all cookies generated to hide SIP URIs in a custom tag appended to the header. Setting the use url parameter field to:

- from-to only affects the behavior of the SIP NAT function's cookies in the From and To headers.
- all affects all SIP headers processed by the SIP NAT function

The cookie is the following:

[;url-parameter]=[host nat tag][encrypted 13-byte host IP][encrypted 13-byte maddr IP]

where:

- [;url-parameter] refers to the SIP NAT function's parameter name field.
  This cookie type is associated with the all and from-to field value options of the SIP NAT function's use url parameter field.
- [host nat tag] refers to the SIP NAT function's host NAT tag field.
- [encrypted 13-byte host IP] refers to the host IP encryption.
- [encrypted 13 byte maddr IP (if present)] refers to the maddr IP encryption, if it exists.

With a host NAT tag of ACME- and a parameter name of acme_param, the following SIP-URI:

```
sip:6175551212@192.168.1.100
```

might be translated into the following:

```
sip:6175551212@172.16.1.10;acme_param=ACME-pfi1s7n2pstna.
```

tel URL

The SIP NAT function cookie is used when devices in your network are strict about the context portion of SIP messages regarding the conversion of tel URLs. This cookie for the tel URL parameter portion of a SIP message is the following:

```
tel URL parameter-[13-byte host IP][13 byte optional maddr IP]domain suffix
```

where:

- tel URL parameter refers to the SIP NAT function's use url parameter.
  This cookie type is associated with the use url parameter's phone field value for the SIP NAT.
- [13-byte host IP] refers to the host IP encryption.
- [13 byte optional maddr IP] refers to the maddr IP encryption, if it exists.
- domain suffix refers to the SIP NAT function's domain suffix field.

Configuration Overview

Configuring the SIP NAT function falls into two areas, the SIP NAT interface parameters and the SIP NAT policies.

SIP NAT Interface

The following tables lists the SIP NAT function interface parameters you need to configure on the Oracle USM (USM).

Parameter	Description
realm ID	Name of the external realm. The realm ID must be unique; no two SIP NATs can have the same realm ID. This realm ID must also correspond to a valid realm identifier entered when you configured the realm.
external proxy address	IPv4 address of the SIP element (for example, a SIP proxy) in the external network with which the USM communicates. Entries must follow the IP address format.
external proxy port	UDP/TCP port of the SIP element (for example, a SIP proxy) in the external network with which the USM communicates. Minimum value is 1025, and maximum value is 65535. Default is 5060.
external address	IPv4 address on the media interface in the external realm. Enter a value that ensures any packet with an external address value as its destination address is routed to the USM through the media interface connected to or routable from the external realm. Entries must follow the IP address format. To specify whether the external realm referenced in this field is private or public, configure the SIP config's NAT mode.

Parameter	Description
home address	IPv4 address on the media interface in the home realm. Enter a value that ensures any packet with a home address value as its destination address must be routed to the USM through the media interface connected to or routable from the home realm. Entries must follow the IP address format. The value entered in this field must be different from the IP address value of the home realm's network interface element. The home realm network interface is associated with this SIP NAT by its realm ID and the realm's identifier and network interface value you entered when you configured the realm. The realm's network interface identifier value corresponds to this SIP NAT's realm ID, the SIP config's home realm ID, and the media manager's home realm ID.
home proxy address	Sets the IP address for the home proxy (from the perspective of the external realm). By default, this field is empty. An empty home proxy address field value signifies that there is no home proxy, and the external address will translate to the address of the USM 's SIP proxy. Entries must follow the IP address format.
home proxy port	Sets the port number for the home realm proxy. Value can be set to zero (0). Minimum is 1025 and maximum is 65535. Default is 5060.
route home proxy	Whether to route all inbound requests for the SIP NAT to the home proxy. enabled adds route if Request-URI is not the USM disabled does not route inbound requests to the home proxy forced always adds route

SIP NAT Function Policies

The following tables lists the SIP NAT function policy parameters you need to configure on the Oracle USM (USM).

Parameter	Description
domain suffix	Domain name suffix of the external realm. The domain name suffix refers to and must conform to the hostname part of a URI. In combination with the user NAT tag and host NAT tag values, this value is used to help the USM identify an encoded URI that it needs to translate when moving between public and private realms. This suffix is appended to encoded hostnames that the SIP NAT function creates. For example, if the encoded hostname is ACME-abc123 and the domain-suffix value is .netnetsystem.com, the resulting FQDN will be ACME-abc123.netnetsystem.com.
address prefix	Defines which IPv4 address prefixes from incoming messages require SIP-NAT encoding (regardless of the realm from which these messages came).
tunnel redirect	Controls whether Contact headers in a 3xx Response message received by the USM are NATed when sent to the initiator of the SIP INVITE message.

Parameter	Description
use url parameter	Establishes whether SIP headers will use the URL parameter entered in the parameter name for encoded addresses that the SIP NAT function creates. Also, if SIP headers will be used, which type of headers will use the URL parameter. For example, all headers or just the From and To headers. Enumeration field.
parameter name	Indicates the name of the URL parameter when use url applies. This field value will be used in SIP NAT encoding addresses that have a use url parameter value of either from-to or all.
user NAT tag	Identifies the prefix used when an address is encoded into the username portion of user@host;name=xxxx; where name = parameter name. The user NAT tag values can consist of any characters that are valid for the userinfo part of a URI. In combination with the domain suffix and host NAT tag field values, this value is used to help the USM identify an encoded URI that it needs to translate when moving between public and private realms.
host NAT tag	Identifies the prefix used when encoding an address into the hostname part of the URI or into a URL parameter. The host NAT tag values refer to domain labels and can consist of any characters that are valid for the hostname part of a URI. In combination with the domain suffix and user NAT tag values, this value is used to help the USM identify an encoded URI that it needs to translate when moving between public and private realms.
headers	Lists the SIP headers to be affected by the USM SIP NAT function. The URIs in these headers will be translated and encrypted, and encryption will occur according to the rules of this SIP NAT.

SIP NAT Function Configuration

To configure the SIP NAT function on an Oracle USM (USM):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-nat** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-nat
   ORACLE(sip-nat)#
   ```

4. **realm-ID**—Enter the name of the realm you want to identify as the external realm.

   The name you use as the realm ID must be unique. No two SIP NAT functions can have the same realm ID. Also, this value must correspond to a valid identifier entry already configured for the realm.

5. **domain-suffix**—Enter the domain suffix to identify the domain name suffix of the external realm. The domain suffix must begin with a (.) dot.

   The domain name suffix refers to and must conform to the hostname part of a URI. For example:

   ```
   .netnetsystem.com
   ```

The domain suffix is appended to encoded hostnames that the SIP NAT function creates. For example, if the encoded hostname is ACME-abc123, the resulting FQDN is ACME-abc123.netnetsystem.com.

6. **external-proxy-address**—Enter the external proxy address to identify the IPv4 address of the SIP element (for example, a SIP proxy) in the external network with which the USM communicates.

   Enter the value in the IP address format. For example:

   ```
   192.168.11.200
   ```

7. **external-proxy-port**—Enter the external proxy port value to identify the UDP/TCP port of the SIP element (for example, a SIP proxy) in the external network with which the USM communicates. The default is **5060**. The valid range is:

   - Minimum—1025
   - Maximum—65535

8. **external-address**—Enter the external address, which is an IPv4 address on the media interface in the external realm.

   Enter the value in the IP address format. For example:

   ```
   192.168.11.101
   ```

   This value must be such that any packet with an external address value as its destination address is routed to the USM through the media interface connected to or routable from the external realm.

9. **home-address**—Enter the home address, which is an IPv4 address on the network interface in the home realm. This value must be such that any packet with a home address value as its destination address must be routed to the USM through the media interface connected to or routable from the home realm.

   Enter the value in the IP address format. For example:

   ```
   127.0.0.10
   ```

   The value entered in this field **must be different** from the IP address value of the home realm's network interface element.

   The home realm network interface is associated with this SIP NAT by its realm ID and the realm's identifier and network interface value you entered when you configured the realm. The realm's network interface identifier value corresponds to this SIP NAT's realm ID, the SIP config's home realm ID, and the media manager's home realm ID.

10. **home-proxy-address**—Enter the home proxy address to set the IP address for the home proxy (from the perspective of the external realm).

    By default, this field is empty. No home proxy address entry signifies there is no home proxy, and the external address will translate to the address of the USM's SIP proxy.

    Enter the value in the IP address format. For example:

    ```
    127.1.0.10
    ```

11. **home-proxy-port**—Enter the home proxy port to set the port number for the home realm proxy. The default value is **0**. The valid range is:

    - Minimum—0, 1025
    - Maximum—65535

12. **route-home-proxy**—Optional. Enable or disable requests being routed from a given SIP-NAT to the home proxy. The default value is **disabled**. The valid values are:
    - **enabled**—All inbound requests for a specific SIP NAT are routed to the home proxy
    - **disabled**—All inbound requests are not routed through the home proxy.
    - **forced**—The Request is forwarded to the home proxy without using a local policy.
13. **address-prefix**—Optional. Indicate the IPv4 address prefix from incoming messages that requires SIP NAT function encoding (regardless of the realm from which these messages came).

    > **Note:**
    >
    > This value overrides the value set in the realm's address prefix field.

    This field's format incorporates an IPv4 address and number of bits in the network portion of the address. For example, a Class C address has a 24-bit network part. The address prefix for 101.102.103.x would be represented as 10.102.103.0/24.

    The default value is an asterisk (*). When you enter this value or do not enter a value, the realm's address prefix value is used.
14. **tunnel-redirect**—Set to one of the following values to indicate whether certain headers in a 3xx Response message received by the USM are NATed when sent to the initiator of the SIP INVITE message. The default is **disabled**. The valid values are:
    - **enabled**—Certain headers in a 3xx Response message are NATed.
    - **disabled**—Certain headers in a 3xx Response message are not NATed.
15. **use-url-parameter**—Establish whether SIP headers will use the URL parameter (configured in the next step) for encoded addresses created by the SIP NAT function. If SIP headers will be used, this value identifies which types of headers will use the URL parameter. The default value is **none**. The available values include:
    - **none**—No headers will use the URL parameter for address encoding.

      The following example illustrates the functionality of an USM using a use url parameter value of none:

      ```
      sip: 1234@1.2.3.4 is translated into sip: 1234-acme-xxxx@5.6.7.8
      ```

      where -acme-xxxx is a cookie and xxxx is the encoded version of 1.2.3.4.
    - **from-to**—From and To headers will use the URL parameter for address encoding

      The following example illustrates the functionality of a USM using a use url parameter value of none:

      ```
      sip: 1234@1.2.3.4 is translated into sip: 1234@5.6.7.8; pn=acme-xxxx
      ```

      where -acme-xxxx is a cookie and xxxx is the encoded version of 1.2.3.4.
    - **all**—All headers will use the URL parameter for address encoding. Acme Packet recommends not using this values because other SIP elements or implementations (other than the Oracle USM) might not retain the URL parameter in subsequent SIP messages that they send to the Oracle USM.
    - **phone**—

If this field is set to either from-to or all, the USM puts the encoded address of the SIP NAT into a URL parameter instead of using the encoding name inside the userinfo part of the address.

16. **parameter-name**—If you have configured the **use-url-parameter** with the from-to or all value, you need to indicate the hostname prefix.

    The parameter name value is used in SIP NAT encoding addresses that have the use url parameter values of from-to or all.

17. **user-NAT-tag**—Enter a value to identify the username prefix used for SIP URIs. The values you can use can include any characters valid for the userinfo part of a URI. This should be made unique for each realm and SIP NAT function.

    The default value is **-acme-**.

    In combination with the domain suffix and host NAT tag values, this value is used to help the USM identify an encoded URI that it needs to translate when moving between public and private realms.

18. **host-NAT-tag**—Enter a value for the host NAT tag field to identify the hostname prefix used for SIP URIs. The value refers to domain labels and can include any characters valid for the hostname part of the URI. This should be made unique for each realm and SIP NAT function.

    The default value is **ACME-**.

    In combination with the domain suffix and user NAT tag values, this value is used to help the USM identify an encoded URI that it needs to translate when moving between public and private realms.

19. **headers**—List the SIP headers you want affected by the SIP NAT function. The URIs in these headers are translated and encrypted, and encryption occurs according to the SIP NAT function rules.

    To enter the full default list, type headers, followed by a Space and -d, then press Enter.

    You can also insert the following tags in SIP NAT headers if you want to replace FQDNs with next hop or SIP interface IP addresses:

    - fqdn-ip-tgt: replaces the FQDN with the target address
    - fqdn-ip-ext: replaces the FQDN with the SIP NAT external address

      Enter the tag using the following format:

      <header-name>=<tag>

      For example:

      To=fqdn-ip-tgt

      The FQDN in a To header is replaced with the target IP address.

      You can insert the following tags to apply NAT treatment to a From header in an INVITE when the gateway sends it into the home realm.

    - ip-ip-tgt: replaces any IP address in the From header with the next hop target
    - ip-ip-ext: replaces any IP address in the From header with the USM's external address

      To view all SIP NAT function parameters, enter a **?** at the system prompt. The following example shows SIP NAT configuration for peering network.

```
sip-nat
        realm-id                        peer-1
        domain-suffix                   .p1.acme.com
```

```
        ext-proxy-address               192.168.11.200
        ext-proxy-port                  5060
        ext-address                     192.168.11.101
        home-address                    127.0.0.10
        home-proxy-address              127.1.0.10
        home-proxy-port                 5060
        route-home-proxy                enabled
        address-prefix                  *
        tunnel-redirect                 disabled
        use-url-parameter               none
        parameter-name
        user-nat-tag                    -p1-
        host-nat-tag                    P1-
        headers                         Call-ID Contact From Join Record-
Route
                                        Refer-To Replaces Reply-To Route

To Via
                                        f i m r t v
```

SIP Realm Bridging

This section explains how to configure the internal routing among realms known as realm bridging. Realm bridging lets you cross-connect SIP interfaces. You can use one of the following two methods for bridging realms:

- local policy bridging: use this method to enable dynamic internal routing between realms if your SIP interfaces do not have the SIP NAT function applied.
- SIP NAT bridging: use this method if your SIP interfaces have the SIP NAT function applied.

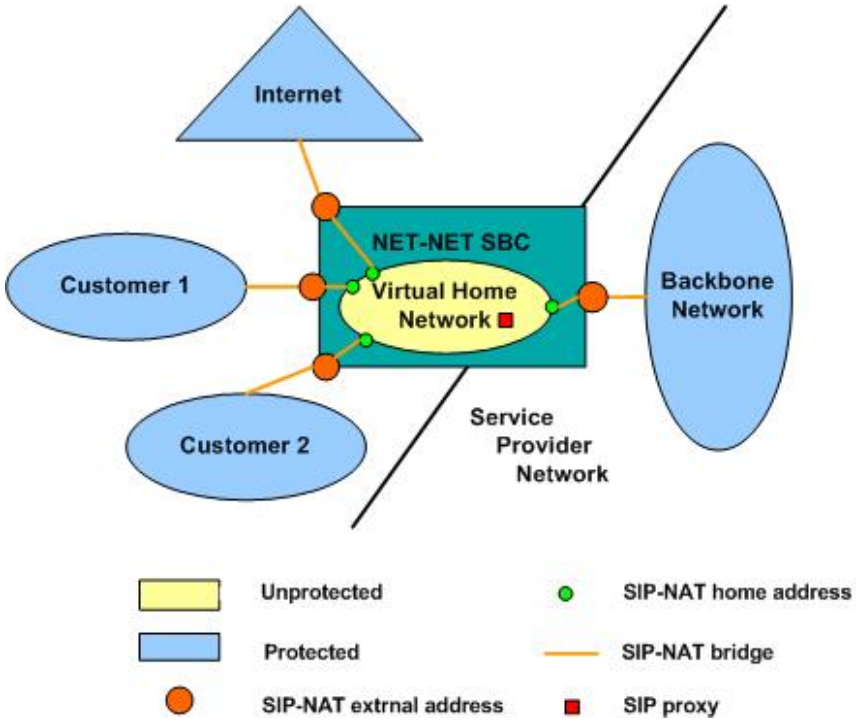
About SIP NAT Bridging

Each SIP NAT has a presence in two realms, trusted and untrusted. The SIP NAT bridge is the conduit for packages in and out of the home realm. It creates a bridge between realms by providing address translations; removing all references to the original IP addressing from the packets sent to the destination network.

With the SIP NAT bridge, an untrusted (or public) home network can reside within the Oracle USM, while the other entities (the backbone network, the Internet, or customer networks) are all trusted (or private). One of the primary functions of the SIP NAT bridge is to protect networks from one another so that address bases can remain hidden. Using a SIP NAT bridge, no one network has direct access to the data of other networks.

Establishing a SIP NAT bridge lets you route every SIP Request message through the backbone. Without using this functionality, it would appear as though all messages/sessions were coming from the Oracle USM's SIP proxy (the SIP server that receives SIP requests and forwards them on behalf of the requestor).

The following diagram illustrates this unprotected (or public) and protected (or private) division.



SIP NAT Bridge Configuration Scenarios

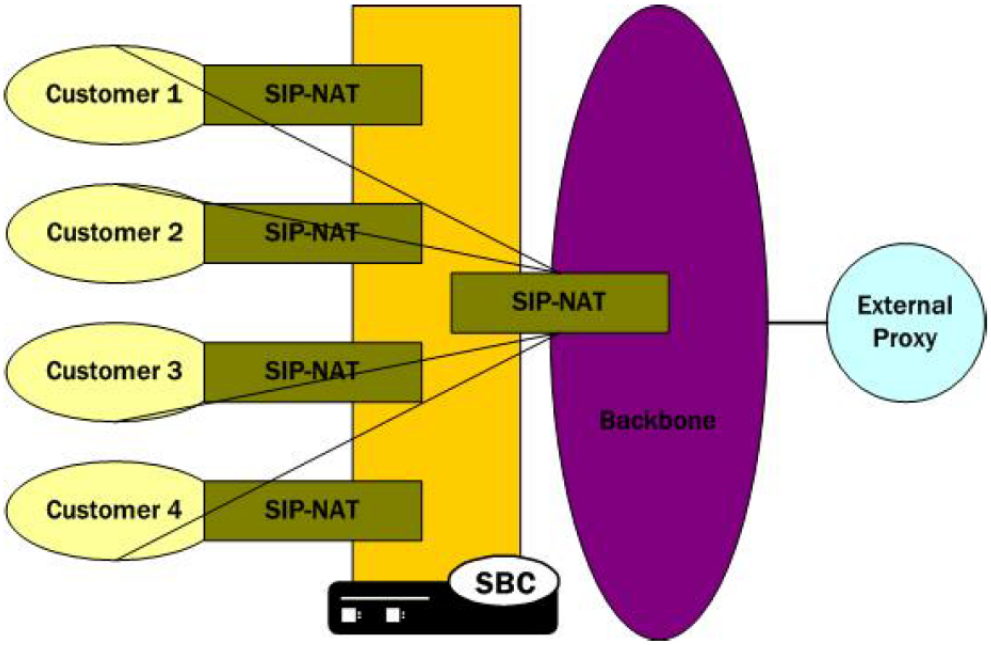
You can configure the SIP NAT bridge functionality in a many-to-one or a one-to-one relationship. For example, multiple customer SIP NATs can be tied to a single backbone SIP NAT, or a single customer SIP NAT can be tied to a single backbone SIP NAT.

You might need to use several SIP NATs on the customer side while using only one on the backbone side in a many-to-one relationship. Or you might configure one SIP NAT on the backbone side for every one that you configure on the customer side in a one-to-one relationship.

You can route all customer side SIP NAT requests to the corresponding backbone SIP NAT regardless of the Request URI. If a request arrives from the customer network with a Request URI that does not match the customer SIP NAT external address or the local policy that would route it to the backbone SIP NAT; the route home proxy value is used.

Many to One Configuration

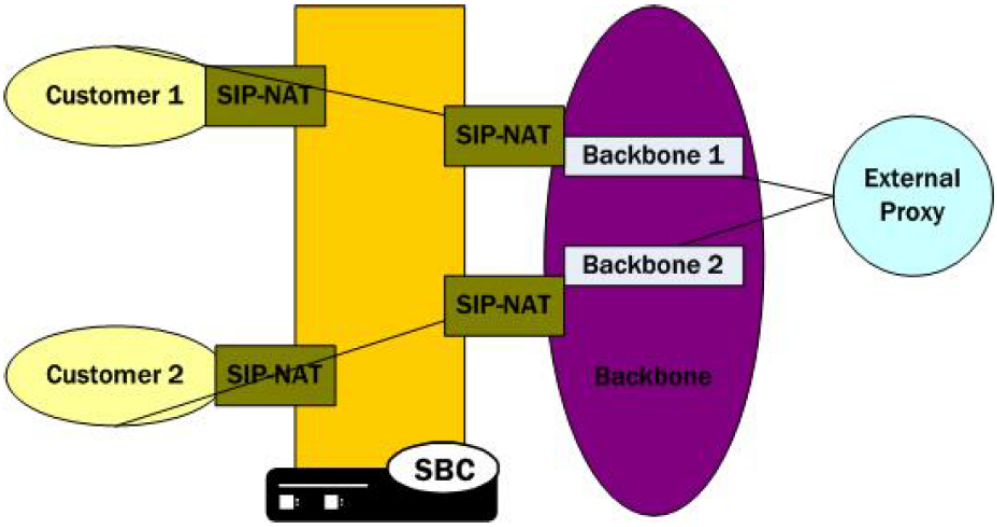
In the many-to-one scenario, multiple customer SIP NATs are tied to a single backbone SIP NAT. The following diagram illustrates the many-to-one SIP NAT bridge configuration.



One-to-One Configuration

In the one-to-one scenario, a single customer SIP NAT is tied to a single backbone SIP NAT. On the backbone SIP NAT side, you configure the home proxy address to match the home address of the customer SIP NAT. On the customer side, you configure the home proxy address to match the home address of the backbone SIP NAT.

The following diagram illustrates the one-to-one SIP-NAT bridge configuration.



SIP NAT Bridge Configuration

You create a bridge between SIP NATs by pointing them at one another. You point the SIP NATs at each other by configuring the home address and home proxy address to create the

bridge. In addition, you can configure the route home proxy on the customer's side of a SIP NAT to force all requests to be routed to the corresponding backbone SIP NAT, regardless of the Request URI. You need to force requests when elements in the customer's network send requests with a Request URI that does not match the customer's SIP NAT external address. Or when the Request URI does not match a local policy element that would route the requests to the backbone SIP NAT.

You also need a home network to create a SIP NAT bridge. If you do not have a real home network, you need to create a virtual one. You also need to configure instances of the SIP NAT to create the SIP NAT bridge within your network.

Creating a Virtual Home Network

A virtual home network is a home network that resides entirely within the Oracle USM, as does a real home network. The difference between the two is the real home network also has a physical connection to the Oracle USM.

The internal home realm/network is usually configured with addresses within the special loopback range (127.0.0.0/8) as described in RFC 3330. This applies to the SIP port addresses for the home realm's SIP interface, and all home addresses for SIP NATs. The address 127.0.0.1 should not be used because it conflicts with the default loopback interface setup by the system for inter-process communication.

To create a virtual home network:

1. Set the name and subport ID of the network interface associated with the home realm element to lo0:0.
2. To enable the SIP proxy to listen for messages on the virtual home realm, configure the home realm ID. It must correspond to the realm's identifier, in which you set the network interface subelement to point to the appropriate network interface element.

   The following table lists the field values you need to set when you are using SIP NAT bridge functionality and you do not have a real home network.

Configuration Element		Sample Values
realm configuration	identifier	home
	network interfaces	lo0:0
	address prefix	127.0.0.0/8
SIP configuration	home realm ID	home
	SIP ports address	127.0.0.100

Many-to-One Configuration

To configure many-to-one:

1. For the backbone SIP NAT, ensure the home proxy address field is blank.
2. For the customer side SIP NAT:

   Set the home address to match the home address of the customer.

   Set the home proxy address to match the backbone SIP NAT home address.

   Set route home proxy to forced.

The following table lists the field values you need to set to create a many-to-one SIP NAT bridge.

SIP NAT Entity	Field	Sample Values
Backbone SIP NAT	home address	IPv4 address of the home realm. For example: 127.0.0.120
	home proxy address	IPv4 address of the home proxy from the perspective of the external realm. For a backbone SIP NAT, leave blank.
Customer SIP NAT	home address	127.0.0.120
	home proxy address	127.0.0.110
	route home proxy	forced

One-to-One Configuration

In the one-to-one scenario, a single customer SIP NAT is tied to a single backbone SIP NAT. The home proxy address field value of the backbone SIP NAT must match the home address of the customer SIP NAT. On the customer side, the home address of the customer SIP NAT should be defined as the home address of the customer, the home proxy address field value should match the home address of the backbone SIP NAT, and route home proxy should be set to forced.

The following table lists the field values you need to set to create a one-to-one SIP NAT bridge.

SIP NAT Entity	Field	Sample Values
Backbone SIP NAT	home address	IPv4 address of the home realm. For example: 127.0.0.110
	home proxy address	IPv4 address of the home proxy from the perspective of the external realm. 127.0.0.120
Customer SIP NAT	home address	127.0.0.120
	home proxy address	127.0.0.110
	route home proxy	forced

Shared Session Agent

Usually, the same set of servers (the external proxy) is used for all SIP NATs to the backbone network. In order to support redundant servers in the backbone of a SIP NAT bridge, the original egress realm as determined by the incoming Request URI needs to be retained after a local policy lookup.

When a request arrives at the Oracle USM, it determines the matching (target) session agent and, after the local policy is examined, sets the new outbound session agent to the one from the selected target.

If the target session agent's realm is set to *, the Oracle USM retains the original session agent's realm ID. Because the target session agent does not have a realm ID defined, the original egress realm is retained.

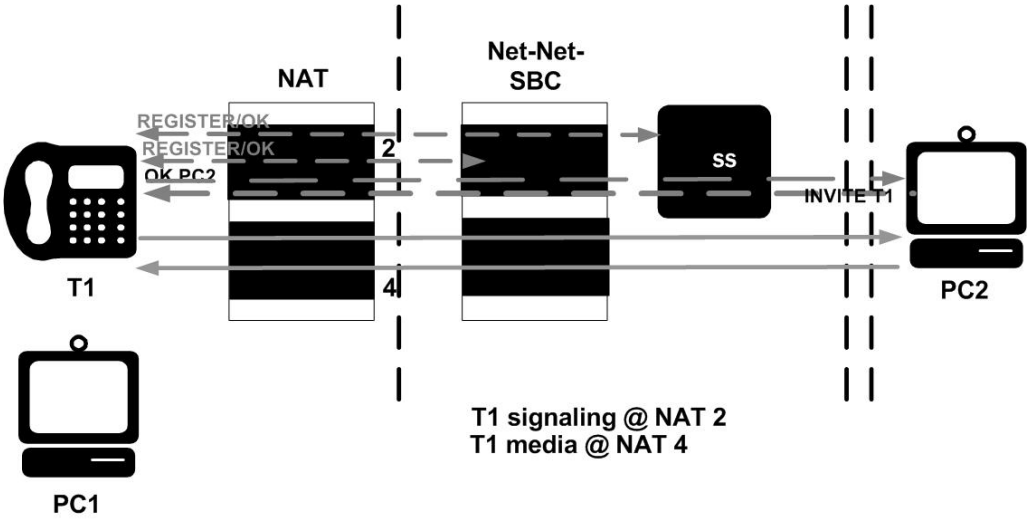
SIP Hosted NAT Traversal (HNT)

This section explains how to configure SIP Hosted Network Address Translation (HNT) traversal. SIP HNT lets endpoints behind a NAT/firewall device send and receive signaling and media using the Oracle USM as a relay.

About SIP HNT

SIP HNT is a technique the Oracle USM uses to provide persistent reachability for SIP UAs located in private Local Area Networks (LANs) behind Nat/firewall devices. It relies on frequent, persistent messaging to ensure that the binding on the intermediary NAT device is not torn down because of inactivity. HNT does not require support for the NAT in the SIP endpoint.

The following diagram illustrates SIP HNT traversal.z



The Oracle USM's HNT function allows endpoints located behind NATs to communicate; providing means to traverse NATs. The Oracle USM interacts with endpoints (using SIP) to allow persistent inbound and outbound signaling and media communications through these NATs.

The Oracle USM automatically detects when an intermediate NAT exists between the UA and the Oracle USM by comparing the Layer 3 IP address of a REGISTER message with the IP address indicated within the UA. The Oracle USM sends signaling responses to the address and port that the request came from, rather than the address and port indicated in the request. The Via header in the request message indicates where the response should be sent.

Using HNT with Existing NAT Device

For network architectures in which premise devices and endpoints reside behind an existing NAT device, the Oracle USM's HNT function allows these premise NATs to be traversed without requiring an upgrade to the premise equipment, the deployment and management of additional premise-based hardware or software, or any NAT device configuration changes.

Registering Endpoints

The Oracle USM uses periodic endpoint registration messages to dynamically establish and maintain bindings in the NAT. These bindings keep a signaling port (port that is opened on a firewall to allow traffic to pass through it is a pinhole) open in the NAT that allows the inbound signaled communications to pass through. Using the endpoint registrations, the Oracle USM then maps the Layer 3 (OSI network layer that deals with switching and routing technologies for data transmission between network devices) IPv4 address/port information from the NAT device to the Layer 5 (OSI session layer that deals with session and connection coordination between applications) entity (for example, user name or phone number) behind the NAT so that when an incoming signaling message is received, the Oracle USM sends it to the appropriate address and port on the NAT for the called party.

Establishing Media Flows

During call setup, the ports for bidirectional media flows are established dynamically. Since the media flows also pass through the Oracle USM, it can identify the IPv4 address/port information on the NAT device used for the outgoing media coming from the user name/phone number. The Oracle USM then uses that same NAT's IPv4 address/port information to send incoming media to the correct user name/phone number behind the NAT device.

Prerequisites

In order to achieve HNT, the endpoints involved must be capable of:

- symmetric signaling: sending and receiving SIP messages from the same transport address (IP address or User Datagram Protocol/Transmission Control Protocol (UDP/TCP) port
- symmetric media: sending and receiving Real-Time Transport Protocol (RTP) messages from the same UDP port

These conditions are required to allow signaling and media packets back through the NAT (through the bound external address and port). These packets must come from the address and port to which the outbound packet that created the NAT binding was sent. The NAT sends these inbound packets to the source address and port of the original outbound packet.

When SIP HNT is used, the Oracle USM sends signaling responses to the address and port that the request came from rather than the address and port indicated in the request. The Via header in the request message indicates where the response should be sent.

Keeping the NAT Binding Open

Additional measures are also required to keep the NAT binding open because most NAT bindings are discarded after approximately a minute of inactivity. The Oracle USM keeps the SIP NAT binding open by returning a short expiration time in REGISTER responses that forces the endpoint to send frequent REGISTER requests.

In order to keep the NAT binding open for SIP, the Oracle USM maintains the registration state. When an endpoint first registers, the Oracle USM forwards that REGISTER message on to the real registrar. You can define the real registrar using either of the following methods:

- Configure the SIP config registrar host and registrar port to indicate the real registrar.
- Map the SIP config registrar host and registrar port values to the SIP NAT home proxy address and home proxy port values. Then configure the SIP NAT's external proxy address and external proxy port values to correspond to the real registrar.

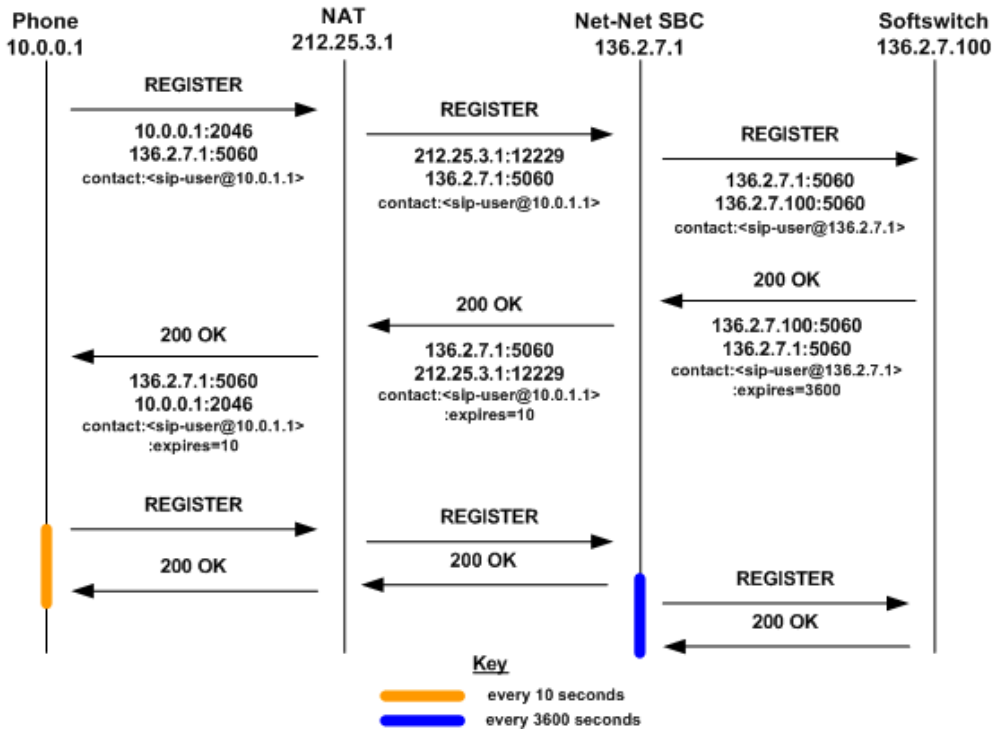
> **Note:**
>
> A registrar can be located in a SIP NAT realm.

When a successful response is received, the Oracle USM caches the registration to memory. This cached registration lives for the length of time indicated by the expiration period defined in the REGISTER response message from the registrar. The response sent back to the endpoint has a shorter expiration time (defined by the SIP config's NAT interval) that causes the endpoint to send another REGISTER message within that interval. If the endpoint sends another REGISTER message before the cached registration expires, the Oracle USM responds directly to the endpoint. It does not forward the message to the real registrar.

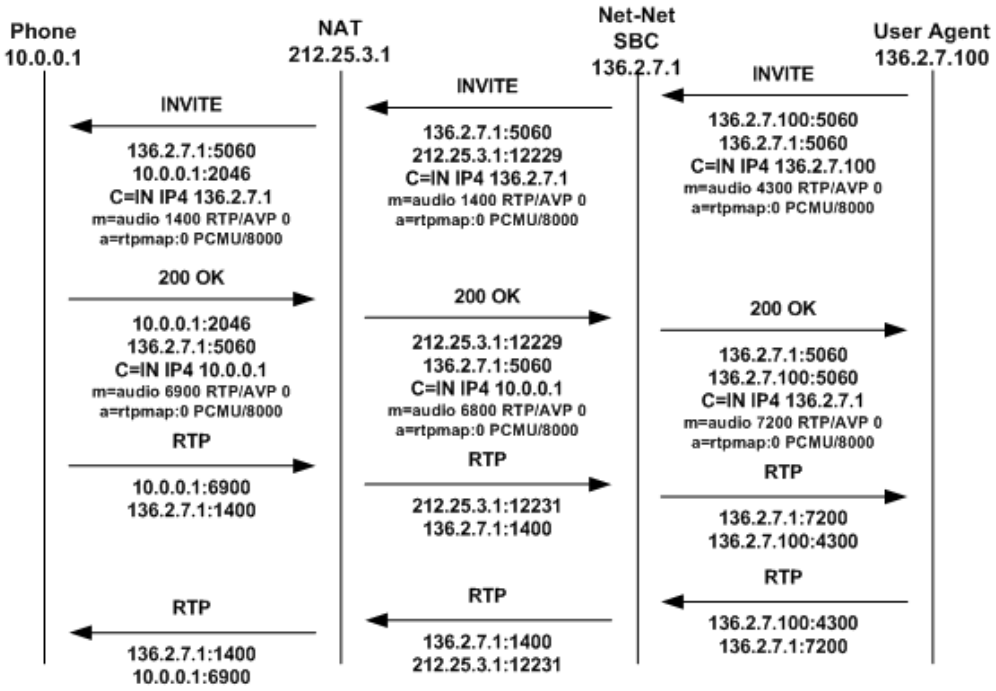
If the cached registration expires within the length of time indicated by the NAT interval, the REGISTER message is forwarded to the real registrar. If the Oracle USM does not receive another REGISTER message from the endpoint within the length of time indicated by the NAT interval, it discards the cached registration.

The Contact Uniform Resource Identifier (URI) in the REGISTER message sent to the registrar by the Oracle USM points at the Oracle USM so that the proxy associated with the real registrar sends inbound requests to the Oracle USM. This way, the inbound requests can be forwarded to the endpoint through the NAT binding.

The following example illustrates the SIP HNT registration call flow for the SIP HNT feature.



The following example illustrates the SIP HNT invitation call flow for the SIP HNT feature.



Working with Multiple Domains

You can use a wildcard (*) with the HNT feature to accommodate multiple domains and to allow the Oracle USM to cache all HNT endpoints. The wildcard functionality is enabled in the SIP config by entering an asterisk (*) in the registrar domain and registrar host fields.

The wildcard allows the use of either a local policy or Domain Name Service (DNS) to resolve the domain name to the correct registrar. Either method can be used to route the Fully Qualified Domain Name (FQDN) when the you enter an asterisk (*) for the register host. An FQDN consists of an unlimited number of domain labels (domain names), each separated by a dot (.). The FQDN can include the top level domain name (for example, acmepacket.com).

In the hostname acme-packet.domainlbl.example100.com, the syntax is as follows:

- acme-packet is a domain label
- domainlbl is a domain label
- example100 is a domain label
- com is the top label

The information configured in a local policy is used before DNS is used. If the next hop destination address (defined in the local policy's next hop field) is an IPv4 address, a DNS server is not needed. A DNS server is needed when the IPv4 address of the next hop destination address is a FQDN or cannot be determined from the Oracle USM's configuration. Even with a configured local policy, the next hop destination address might be an FQDN that requires a DNS lookup.

If the registrar host does not use the wildcard, the Oracle USM always uses the configured address. You can limit the number of endpoints that receive the HNT function. For example, you can use a non-wildcarded registrar domain field value (like acme.com) with a wildcarded registrar host field value.

HNT Configuration Overview

To configure SIP HNT NAT traversal, you need to configure both the SIP interface and the SIP config.

SIP HNT Single Domain Example

The following example shows values entered for the SIP config and SIP interface elements to configure SIP HNT for a single domain and registrar.

- SIP config

Parameter	Sample Value
registrar domain	netnetsystem.com
registrar host	192.168.12.1
registrar port	5060

- SIP interface

Parameter	Sample Value
NAT traversal	always
NAT interval	60
minimum registration expire	200
registration caching	disabled
route to registrar	enabled

SIP HNT Multiple Domain Example

The following example shows values entered for the SIP config and SIP interface elements to configure SIP HNT for a multiple domains and multiple registrars.

- SIP config

Parameter	Sample Value
registrar domain	*
registrar host	*
registrar port	0

- SIP interface

Parameter	Sample Value
NAT traversal	always
NAT interval	60
minimum registration expire	200
registration caching	disabled
route to registrar	enabled

HNT Configuration

To configure a SIP interface on the Oracle USM (USM):

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure physical interface parameters. To view all SIP interface parameters, enter a **?** at the system prompt.

4. **nat-traversal**—Define the type of HNT enabled for SIP. The default value is **none**. Available values include:

   - **none**—Disables the HNT feature for SIP (default value)
   - **rport**—SIP HNT function only applies to endpoints that include the rport parameter in the Via header and the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address.
   - **always**—SIP HNT applies to requests when the sent-by of the topmost VIA matches the Contact-URI host address, both of which must be different from the received Layer 3 address. (Even when the rport parameter is not present.)

5. **nat-interval**—Set the expiration time in seconds for the USM's cached registration entry for an HNT endpoint. The default value is **30**. The valid range is:

   - Minimum—0
   - Maximum—999999999

     Acme Packet recommends setting the NAT interval to one-third of the NAT binding lifetime. A NAT binding lifetime is the network connection inactivity timeout. The value is configured (or hardwired) in the NAT device (firewall). This timer is used to prevent the NAT device from keeping an unused port open.

6. **registration-caching**—Enable for use with all UAs, not just those that are behind NATs. By default, this field is set to **disabled**. If enabled, the USM caches the Contact header in the UA's REGISTER request when it is addressed to one of the following:

   - USM
   - registrar domain value
   - registrar host value

     The USM then generates a Contact header with the USM's address as the host part of the URI and sends the REGISTER to the destination defined by the registrar host value.

     Whether or not SIP HNT functionality is enabled affects the value of the user part of the URI sent in the Contact header:

- **enabled**—The USM takes the user part of the URI in the From header of the request and appends a cookie to make the user unique. A cookie is information that the server stores on the client side of a client-server communication so that the information can be used in the future.
- **disabled**—The user part of the Contact header is taken from the URI in the From header and no cookie is appended. This is the default behavior of the Oracle USM.

  When the registrar receives a request that matches the address-of-record (the To header in the REGISTER message), it sends the matching request to the USM, which is the Contact address. Then, the v forwards the request to the Contact-URI it cached from the original REGISTER message.

7. **min-reg-expire**—Set the time in seconds for the SIP interface. The value you enter here sets the minimum registration expiration time in seconds for HNT registration caching. The default value is **300**. The valid range is:
   - Minimum—1
   - Maximum—999999999

   This value defines the minimum expiration value the USM places in each REGISTER message it sends to the real registrar. In HNT, the USM caches the registration after receiving a response from the real registrar and sets the expiration time to the NAT interval value.

   Some UAs might change the registration expiration value they use in subsequent requests to the value specified in this field. This change causes the USM to send frequent registrations on to the real registrar.
8. **registration-interval**—Set the USM's cached registration entry interval for a non-HNT endpoint. Enter the expiration time in seconds that you want the USM to use in the REGISTER response message sent back to the UA. The UA then refreshes its registration by sending another REGISTER message before that time expires. The default value is **3600**. The valid range is:
   - Minimum—1

     A registration interval of zero causes the USM to pass back the expiration time set by and returned in the registration response from the registrar.
   - Maximum—999999999

     If the expiration time you set is less than the expiration time set by and returned from the real registrar, the USM responds to the refresh request directly rather than forwarding it to the registrar.

   > **Note:**
   >
   > With registration caching, there is no NAT; therefore, a short registration interval causes the UA to send excess REGISTER messages.

   Although the registration interval applies to non-HNT registration cache entries, and the loosely related NAT interval applies to HNT registration cache entries, you can use the two in combination. Using a combination of the two means you can implement HNT and non-HNT architectures on the same USM. You can then define a longer interval time in the registration interval field to reduce the network traffic and load caused by excess REGISTER messages because there is no NAT binding to maintain.

9. **route-to-registrar**—Enable routing to the registrar to send all requests that match a cached registration to the destination defined for the registrar host; used when the Request-URI matches the registrar host value or the registrar domain value, not the SBC's address. Because the registrar host is the real registrar, it should send the requests back to the USM with the USM's address in the Request-URI. The default value is **disabled**. The valid values are:

   - enabled | disabled

     For example, you should enable routing to the registrar if your network uses a USM and needs requests to go through its service proxy, which is defined in the registrar host field.

Global SIP Configuration

To configure the SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. **registrar-domain**—Optional. Define the domain to match against the host part of a URI to determine if a request is addressed to the registrar. If there is a match, the registration caching, NAT traversal, and route to registrar parameter values for the SIP interface are applied to the request. By default, this field remains empty. Available values are:

   - an asterisk (*) to specify the values apply to all requests.
   - any alphanumeric character or any combination of alphanumeric characters. For example, acme1.com.

     A hostname consists of any number of domain labels, separated by dots (.), and one top label. A top label is the last segment of the hostname. It must start with an alphabetical character. After the first character, a top label can consist of any number or combination of alphanumeric characters, including those separated by dashes. The dash must be preceded and followed by alphanumeric characters. A single alphabetical character is the minimum requirement for a hostname field (for example, c to indicate .com).

     When the REGISTER message's Request-URI has an FQDN, it is matched against the registrar domain's value to determine if the message needs to be forwarded to the registrar port on the registrar host. The registrar domain's value is also used when route to registrar is set to enabled, to determine if a request needs to be forwarded to the registrar.

     Only the right-hand part of the domain name in the Request-URI needs to match the registrar domain value. For example, acme3.acmepacket.com matches acmepacket.com. However, the entire domain label within the domain name must

match. For example, the domain label “acme3.acmepacket.com” would not match packet.com.

5. **registrar-host**—Define the address of the registrar for which requests for registration caching, NAT traversal, and router to registrar options apply. You can use a specific hostname, a IP address, or a wildcard (*):

   - an asterisk (*) indicates normal routing (local policy, DNS resolution, and so on) is used to determine the registrar’s address.
   - hostname: can consist of any alphanumeric character or any combination of alphanumeric characters (for example, acme1.com). The hostname can consist of any number of domain labels, separated by dots (.), and one top label. You can use the minimum field value of a single alphabetical character to indicate the top label value (for example, c to indicate .com).
   - IPv4 address: must follow the dotted notation format. Each of the four segments can contain a numerical value between zero (0) and 255. For example, 192.168.201.2. An example of a invalid segment value is 256.

     By default, the registrar host field remains empty.

6. **registrar-port**—Set the SIP registrar port number. The SIP registrar server configured in this and the registrar host field is the real registrar. Or the values entered in those fields map to the home proxy address and home proxy port of the SIP NAT with external proxy address and external proxy port values that correspond to the real registrar. The default value is 0. The valid range is:

   - Minimum—0, 1025
   - Maximum—65535

   The following example shows the values for a single domain and registrar configuration.

```
sip-config
        state                         enabled
        operation-mode                dialog
dialog-transparency        disabled
        home-realm-id                 acme
        egress-realm-id
        nat-mode                      Public
        registrar-domain
        registrar-host
        registrar-port                0
        init-timer                    500
        max-timer                     4000
        trans-expire                  32
        invite-expire                 180
        inactive-dynamic-conn         32
        red-sip-port                  1988
        red-max-trans                 10000
        red-sync-start-time           5000
        red-sync-comp-time            1000
        last-modified-date            2005-03-19 12:41:28
```

Endpoint-Initiated Keep-Alives

The Oracle USM provides three Keep-Alive algorithms designed to generate sufficient traffic flow to maintain NAT (Network Address Translation) bindings. Prior releases provided a single Keep-Alive algorithm that is described in the SIP Hosted NAT Traversal (HNT) section of the

SIP Signaling Services chapter of the *ACLI Configuration Guide*. This method is SD-based and requires no explicit participation by the SIP endpoint. Instead the SD manipulates SIP registration requests and responses to spoof the endpoint — generating frequent and extraneous registration requests that generate a periodic traffic flow to maintain existing NAT bindings.

Unlike the current SD-centric method, the new methods require the active participation of the SIP endpoint. With these methods the SIP endpoint initiates a Keep-Alive negotiation with the SD that produces a periodic request/response message sequence which also generates sufficient traffic flow to maintain NAT bindings.

The new methods are based upon the following RFCs.

RFC 5626, *Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)*, which, in Section 3.5.2, provides a framework for the implementation of an endpoint-initiated Keep-Alive on both TCP and UDP connections and provides two approaches to generating endpoint-initiated Keep-Alive traffic flows. One approach described in Section 4.4.1 of the RFC is restricted to connection-oriented transport protocols such as TCP. It defines an end-point initiated ping which requires an SD pong in response. A second approach described in Section 4.4.2 of the RFC is restricted to connectionless transport protocols such as UDP. It defines an endpoint-initiated STUN binding request which requires as SD-originated STUN binding response.

RFC 6223, *Indication of support for Keep-Alive*, was initially published as an internet draft (most recently as draft-ietf-sipcore-keep-12.txt). RFC 6223 defines a procedure that enables a SIP endpoint to signal its capability and willingness to send and receive periodic Keep-Alive messages to a device referred to by the RFC as an edge proxy, a role performed by the SD. After receiving such a signal, the SD returns a response indicating its willingness to exchange Keep-Alives, and specifying the interval between Keep-Alive exchanges.

RFC 5389, *Session Traversal Utilities for NAT (STUN)*, defines STUN binding requests and responses in Section 6 of the RFC. Endpoints that support endpoint-initiated Keep-Alives over a UDP connection must be capable of constructing and transmitting a STUN binding request, and receiving and parsing a STUN binding response.

Endpoint-Initiated Keep-Alive Negotiation

Endpoint-initiated Keep-Alive negotiation requires the presence of the keep parameter in the Via header of Registration requests and responses issued and received by a SIP endpoint. As defined in RFC 6223, keep is

A SIP Via header field parameter that a SIP entity can insert in the topmost Via header field that it adds to the request, to explicitly indicate willingness to send keep-alives towards its adjacent downstream SIP entity. A SIP entity can add a parameter value to the keep parameter in a response to explicitly indicate willingness to receive keep-alives from its adjacent upstream SIP entity.

While RFC 6223 allows the presence of the keep parameter in either REGISTER or INVITE requests and responses, the current implementation, is registration-based. keep parameters contained in methods other than REGISTER are ignored.

As shown in the following sample registration exchange, the SIP endpoint signals its willingness to exchange Keep-Alive messages by placing an unvalued keep parameter in the SIP Via header of a REGISTER request.

```
REGISTER sip:512@172.16.101.23:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;keep
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
```

```
Call-ID:1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: "512" <sip:512@172.16.101.38:5070>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
```

The SD strips the keep parameter from the Via header of the REGISTER request, and forwards the request to the Registrar.

```
REGISTER sip:512@192.168.7.32:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 69
Contact: "512" <sip:512@192.168.101.23:5060;transport=udp>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
```

The Registrar indicates successful registration with a 200 OK REGISTER response back to the SD.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@192.168.101.23:5060;transport=udp>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

The SD indicates its willingness to exchange Keep-Alives by assigning a value, which specifies the interval between Keep-Alives, to the keep parameter and re-inserting the parameter and its assigned value into the Via header of the REGISTER response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;keep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@172.16.101.38:5070>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

Connection Oriented Keep-Alives

After the endpoint-initiated Keep-Alive exchange has been negotiated for a TCP connection, the SIP endpoint transmits a periodic ping at intervals specified by the value of the keep parameter. As shown below, the ping consists of carriage return (CR) and line feed (LF) characters.

```
CRLFCRLF
CR = %x0D
LF = %x0A
```

The SD responds with a pong as shown below.

```
CRLF
CR = %x0D
LF = %x0A
```

The request ping and the response pong are transmitted between SIP messages, and cannot be sent in the middle of SIP message. If the TCP connection is secured by TLS, the Keep-Alive requests and responses are transmitted within the TLS encrypted stream.

As specified in Section 4.4.1 of RFC 5626 the SD must respond to a ping within a 10-second interval. If the endpoint fails to receive a valid pong within that interval, it must treat the flow as failed.

Connectionless Keep-Alives

After the endpoint-initiated Keep-Alive exchange has been negotiated for an unreliable UDP connection, the SIP endpoint, acting as a STUN client, transmits a periodic STUN binding request so that the interval between each request is randomly distributed between 80 and 100 percent of the value of the keep parameter. Assuming a parameter value of 20 seconds, for example, the SIP endpoint transmits a STUN binding request at random intervals between 16 and 20 seconds in length.

The STUN binding request has the following format.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0|     STUN Message Type     |         Message Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic Cookie                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Transaction ID (96 bits)                  |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The initial two bits of any STUN message are always 00

STUN Message Type

This 14-bit field contains the message class and method — for a STUN binding request, class is request and method is binding.

Message Length

This 16-bit field contains the length, in octets, of the Attribute section of the STUN binding request. Since the Attribute section is not present in a binding request, this field contains a value of 0.

Magic Cookie

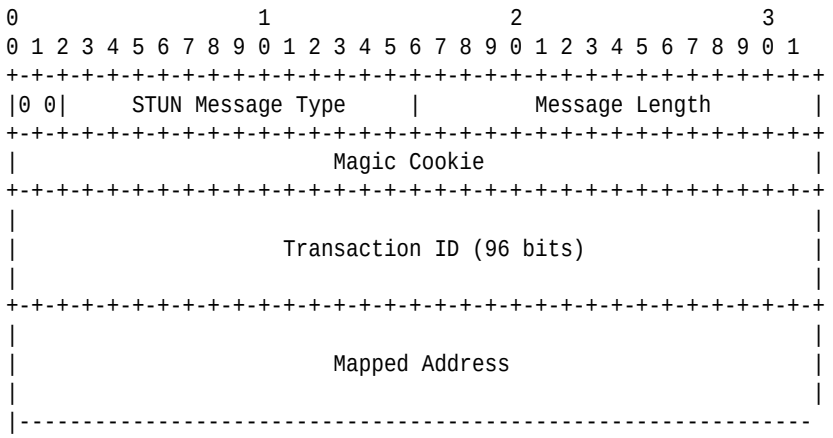
This 16-bit field always contains a value of 0x2112A442.

Transaction ID

This 96-bit field contains a random value that provides a transaction identifier. The value is generated by the STUN client and echoed in the STUN server response.

Upon receipt of a binding request, the SD, acting as a STUN server, transmits a STUN binding response. Receipt of the binding response by the endpoint confirms the UDP connection between the endpoint and the SD, and the viability of NAT bindings in the transmission path.

The STUN binding response has the following format.



The initial two bits of any STUN message are always 00.

STUN Message Type

This 14-bit field contains the message class and method — for a STUN binding response, class is response and method is binding.

Message Length

This 16-bit field contains the length, in octets, of the Attribute section of the STUN binding response. Since the binding response contains a single attribute, MAPPED-ADDRESS, the message length can be either 8 or 16 octets, depending on the IP address type.

Magic Cookie

This 16-bit field always contains a value of 0x2112A442.

Transaction ID

This 96-bit field contains a random value that provides a transaction identifier. The value is generated by the STUN client and echoed by the STUN server.

MAPPED-ADDRESS

This attribute contains the IP address and port number of the proximate NAT device, that is the address translator closest to the SD, that forwarded the STUN request toward the SD. Specific fields within the attribute identify the IP family (Version 4 or 6), the IP address, and port number. The attribute length is 8 octets if it contains an IPv4 address, or 16 octets if it contains an IPv6 address.

Once initiated, endpoint transmission of STUN binding requests and SD responses continue for the duration of the SIP Registration, 1 hour in the sample negotiation, or until the endpoint transmits a new REGISTER request. In the event of such a new request, the endpoint once again indicates its willingness to exchange STUN Keep-Alives with an unvalued keep parameter in the Via header. If endpoint-initiated Keep-Alive renegotiation is not successful, the endpoint must cease the transmission of Keep-Alive messages.

An endpoint failure to issue a timely STUN binding request is not fatal. In the absence of an expected request, the SD takes no action with regard to the UDP connection, or to current calls,

nor does it remove entries from its registration cache. After reception of a tardy STUN request, the SD simply returns a STUN response.

In contrast, however, Section 4.4.2 of RFC 5626 states that an endpoint "considers the flow failed" if (1) it receives a malformed STUN bonding response from the SD, or (2) it fails to receive a timely binding response from the SD. If the endpoint receives a STUN binding response that contains a different MAP Address field from the previous response, It MUST treat this event as a failure on the flow.

Enpoint-Initiated Keep-Alives Configuration

You use the **registrar-keep-alive** attribute, available in SIP Interface configuration mode, to enable endpoint-initiated Keep-Alives on a SIP interface.

1. In Superuser mode, use the following ACLI command sequence to access SIP Interface configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

2. The **registrar-keep-alive** attribute enables endpoint-initiated Keep-Alive on the current SIP interface.

   none — (the default) disables endpoint-initiated Keep-Alive processing

   always — assuming that the endpoint has included the keep parameter in the Via REGISTER request header, forces the SD to place a keep parameter and an assigned in the Via REGISTER response — thus enabling endpoint-initiated Keep-Alive exchange with that endpoint

   bnat — assuming that the endpoint has included the keep parameter in the Via REGISTER request header, forces the SD to place a keep parameter and an assigned value in Via REGISTER response only if the requesting endpoint is identified as being behind an intervening NAT device (based on comparing source IP packet addresses with IP addresses extracted from the SIP request).

   ```
   ORACLE(sip-interface)# registrar-keep-alive bnat
   ORACLE(sip-interface)#
   ```

3. If Keep-Alive is enabled on the current SIP interface (**registrar-keep-alive** is always or bnat), use the **nat-interval** attribute to specify the value of the keep parameter provided by the SD to SIP endpoints served by a UDP connection.

   Allowable values are integers within the range 1 through 999999999 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 30 seconds.

   The SIP endpoint transmits periodic STUN binding requests so that the interval between each request is randomly distributed between 80 and 100 percent of the value of the nat-interval attribute.

   Assuming the default value (30 seconds) the interval between STUN binding requests would vary from 24 to 30 seconds. This default value closely aligns to Section 4.4.2 of RFC 5626, which recommends that the time between each keep-alive request SHOULD be a random number between 24 and 29 seconds.

   ```
   ORACLE(sip-interface)# nat-interval 20
   ORACLE(sip-interface)#
   ```

4. If Keep-Alive is enabled on the current SIP interface (**registrar-keep-alive** is always or bnat), use the **tcp-nat-interval** attribute to specify the value of the keep parameter provided by the SD to SIP endpoints served by a TCP or TCP/TLS connection.

   Allowable values are integers within the range 1 through 999999999 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 90 seconds.

   The SIP endpoint transmits periodic pings at intervals specified by this attribute.

   ```
   ORACLE(sip-interface)# tcp-nat-interval 120
   ORACLE(sip-interface)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete this configuration.

6. If necessary, repeat Steps 1 through 5 to configure Keep-Alives on additional SIP interfaces.

SD-originated Keep-Alive Negotiation

SD-originated Keep-Alives are useful on TCP connections with iOS (iPhone Operating System) clients, such as iPhones or iPads. iOS places inactive clients in a dormant state after a period of time. Once in the dormant state, clients cannot transmit date unless awakened by network traffic. SD-originated Keep-Alives are generated at a sufficient frequency to prevent iOS clients from entering the dormant state.

SD-originated Keep-Alive negotiation is similar to the earlier described negotiation, except for the parameter used. SD-originated Keep-Alive negotiation requires the presence of a well known proprietary parameter in the Via header of Registration requests and responses issued and received by a SIP endpoint. The proprietary parameter can be defined as follows:

A SIP Via header field parameter that a SIP endpoint can insert in the topmost Via header field that it adds to the request, to explicitly indicate willingness to receive Keep-Alives from its adjacent downstream SIP entity. This same SIP endpoint can assign a value to this proprietary parameter in the response to specify the keep-alive interval.

As shown in the following sample registration exchange, the SIP endpoint signals its willingness to receive Keep-Alive messages by placing an rkeep parameter in the SIP Via header of a REGISTER request; the assigned value of 20 specifies the interval between Keep-Alives.

```
REGISTER sip:512@172.16.101.23:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;rkeep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID:1-14400@172.16.101.38
CSeq: 1 REGISTER
Max-Forwards: 70
Contact: "512" <sip:512@172.16.101.38:5070>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
```

The SD strips the rkeep parameter from the Via header of the REGISTER request, and forwards the request to the Registrar.

```
REGISTER sip:512@192.168.7.32:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
```

```
Max-Forwards: 69
Contact: "512" <sip:512@192.168.101.23:5060;transport=udp>;expires:18000
User-Agent: ADTRAN_Total_Access_908e_(2nd_Gen)/A1.02.00.E
Content-Length: 0
Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
```

The Registrar indicates successful registration with a 200 OK REGISTER response back to the SD. The expires parameter in the Contact header grants a registration period of 1 hour (3600 seconds).

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.101.23:5060;branch=z9hG4bK3q43klh3dafekdepnbaqip04e1
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@192.168.101.23:5060;transport=udp>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

After inserting the rkeep parameter and assigned value in the Via header of the REGISTER response, the SD forwards the response to the endpoint The presence of the rkeep parameter signals the SD's willingness to transmit Keep-Alive messages, and the parameter value specifies the exchange frequency in seconds.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 172.16.101.38:5070;branch=dd1;rkeep=20
From: "512" <sip:512@172.16.101.38:5070;transport=UDP>;tag=443322
To: "512" <sip:512@172.16.101.38:5070>;tag=b68b3d53a5a90225609112ff6c211bef.16a6
Call-ID: 1-14400@172.16.101.38
CSeq: 1 REGISTER
Contact: <sip:512@172.16.101.38:5070>;expires=3600
Server: OpenSER (1.3.0-notls (i386/linux))
Content-Length: 0
```

SD-Originated Keep-Alive Format

After the Keep-Alive exchange has been negotiated for a TCP connection, the SD, transmits a periodic ping at a frequency specified by the proprietary parameter (rkeep in the example negotiation). As shown below, the ping consists of carriage return (CR) and line feed (LF) characters.

CRLFCRLF

CR = %x0D

LF = %x0A

pings are transmitted between SIP messages, and cannot be sent in the middle of SIP message. If the TCP connection is secured by TLS, pings are transmitted within the TLS encrypted stream.

SD-Initiated Keep-Alives Configuration

Use the following procedure to configure SD-initiated Keep-Alives.

1. In Superuser mode, use the following ACLI command sequence to access SIP Interface configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. The **options sbc-initiated-keep-alive-name** attribute enables SD-initiated Keep-Alive on the current SIP interface and specifies the name of the proprietary Via header parameter used during the SD-initiated Keep-Alive negotiation.

   Prefix the command with a plus (+) sign to avoid over-writing existing options configuration.

   The following example enables SD-initiated Keep-Alives and identifies the proprietary rkeep header as the header that contains negotiation information.

   > **Note:**
   >
   > Endpoints require prior knowledge of this proprietary header name because the SD lacks the capability to relay configuration data to relevant endpoints.

   ```
   ORACLE(sip-interface)# options +sbc-initiated-keep-alive-name=rkeep
   ORACLE(sip-interface)#
   ```

3. The **option sbc-initiated-keep-alive-interval** attribute specifies the frequency of SD-initiated Keep-Alives.

4. If SD-initiated Keep-Alive is enabled on the current SIP interface (**options sbc-initiated-keep-alive-name** identifies a proprietary Via header parameter used for Keep-Alive negotiations), use the **options sbc-initiated-keep-alive-interval** attribute to specify the default interval between SD-initiated pings.

   Prefix the command with a plus (+) sign to avoid over-writing existing options configuration.

   Allowable values are integers within the range 1 through 999999999 (seconds).

   In the absence of an explicit assignment, this attribute defaults to a value of 30 seconds.

   This default interval will be used only the proprietary parameter proposed by the endpoint lacks a value.

   ```
   ORACLE(sip-interface)# options +sbc-initiated-keep-alive-interval=45

   ORACLE(sip-interface)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete this configuration.

6. If necessary, repeat Steps 1 through 5 to configure SD-initiated Keep-Alives on additional SIP interfaces.

Statistics

The SD does not gather or maintain endpoint-initiated Keep-Alive statistical counts. However, malformed STUN binding requests and responses are included in the count of mal-formed SIP messages. Additionally, if DoS is enabled, STUN binding exchanges are included in endpoint message counts.

SIP Registration Local Expiration

When you deploy multiple Oracle USMs (USM) in series and they have registration caching and HNT configured, registration cache entries might expire prematurely in instances with several devices provisioned with the same address of record (AoR). Now you can configure a SIP interface option to prevent the premature expiration.

When you use registration caching and HNT, the USM adjusts the expiration time it sends to user agents (UAs) in REGISTER responses based on the registration interval you configure. It can be the case that a SIP user has multiple registered contact endpoints at the UA to which a response is sent. If the URI in the Contact contains the UA‘s address and that UA included the Contact in the REGISTER request, then the Contact is seen as exclusively belonging to that UA. In the REGISTER response, this Contact (exclusive to the UA) includes the local expiration time, a time based on the SIP interface configuration’s registration or NAT interval value. Additional Contacts (not exclusive to the UA) in the REGISTER response have the expiration time from the REGISTER response the registrar sent to the USM.

It is this default behavior can cause registration cache entries to expire prematurely in the USM nearest a registrar when multiple USMs are deployed in series. Multiple registering UAs for a single SIP user, for example, might trigger the early expiration. The SIP you can configure an option per SIP interface that causes the USM to send the local registration expiration time in all in the Expires parameter of all Contact headers included in REGISTER responses sent from the SIP interface.

SIP Registration Local Expiration Configuration

You can configure this feature either for the global SIP configuration, or for an individual SIP interface.

To configure SIP registration local expiration for the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. If you are editing an existing configuration, select the configuration so you can enable this feature.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **reg-local-expires** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-config)# options +reg-local-expires
   ```

   If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration’s **options list, you must prepend the new option with a plus sign as shown in the previous example.**

5. Save and activate your configuration.

To configure SIP registration local expiration for an individual SIP interface:

6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

7. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

8. Type **sip-interface** and press Enter. If you are editing an existing configuration, select the one on which you want to enable this feature.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

9. **options**—Set the options parameter by typing **options**, a Space, the option name **reg-local-expires** with a plus sign in front of it, and then press Enter.

```
ORACLE(sip-interface)# options +reg-local-expires
```

If you type options and then the option value for either of these entries **without the plus sign, you will overwrite any previously configured options. In order to append the new option to this** configuration's **options list, you must prepend the new option with a plus sign as shown in the previous example.**

10. Save and activate your configuration.

Simultaneous TCP Connection and Registration Cache Deletion

You can configure the Oracle USM to automatically start a timer when a user deregisters or changes location by registering with a new contact address.

Not all devices tear down TCP connections associated with these old addresses when a user registers with a new contact address.

Registration Cache Deletion Configuration

You can apply **suppress-reinvite** to the sip-interface facing the User Agents whose re-INVITEs are to be responded to locally.

To enable Registration Cache Deletion:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **tcp-conn-dereg**—The delay (in seconds) that is used to determine when to terminate the TCP connection for a user that has been removed from the registration cache or changed location. This feature can be disable by setting the value to zero.

   ```
   ORACLE(session-agent)# tcp-conn-dereg 5300
   ```

5. Save your work.

SIP HNT Forced Unregistration

If you use HNT and experience the issue explained in this section, consider using the Oracle USM (USM) forced unregistration feature. When this feature is enabled and a registration entry for an endpoint expires, the USM notifies the soft switch to remove this binding using REGISTER message. In that REGISTER message, the expires header will be set to 0 and the expires parameter in the Contact header will also be set to 0.

The benefits of using forced unregistration include:

- Leveraging existing HNT configuration to provide near real-time information about the UA's status to the registrar/softswitch
- Preserving resource utilization for the USM and the softswitch by deleting a contact binding that is no longer valid or needed
- Preventing extra bindings from being generated at the softswitch (e.g., in instances when the UA or NAT restart)

This feature applies to:

- HNT endpoints with registration caching enabled by default, and when the **nat-traversal** parameter in the SIP interface configuration is set to always
- non-HNT endpoints with registration caching enabled, when the registration-interval parameter in the SIP interface configuration is used in the expires header sent to the UA in the 200 OK

When to Use Forced Unregistration

For typical HNT use, it is common that the registration interval between the client UA and the Oracle USM (USM) is between 60 and 120 seconds. This differs significantly from the re-registration interval between the USM and the and the registrar, which varies from approximately 30 to 60 minutes.

If the UA fails to refresh its registration, the contact binding at the USM is deleted after the registration expires. This expiration is determined by the expires= header in the 200 OK. The binding at the real registrar will remain intact. This creates a discrepancy between the real state of the UA and state of the softswitch. In the best case scenario, the contact binding expires at the softswitch after a few minutes.

For network management, this discrepancy can be problematic because the service provider would be unaware of the UA's status until the binding expires at the softswitch. This can take a considerable amount of time to happen.

In addition, the USM encodes a cookie in the userinfo of the Contact header in the REGISTER message. This is a function of the source IPv4 address and port from which the request came, i.e., the ephemeral port in the NAT for DSL scenarios. Therefore, additional bindings that remain for long periods of time are created at the registrar if, for example, the:

- UA reboots

- Ethernet link between the UA and the DSL router is lost for over two minutes
- DSL crashes
- DSL/ATM layer between the DSL router

Caution for Using Forced Unregistration

You should use caution when applying SIP HNT forced unregistration for the following reasons:

- It can have an impact on the performance of your Oracle USM and the registrar, especially when you have a large number of HNT endpoints in your configuration that become unavailable simultaneously.
- It is possible that the registrar might become vulnerable to overload in the case where the registrar must authenticate a large number of register messages generated when HNT endpoints are de-registered. It is possible that the cached registration credentials might become "stale" over time (e.g., the nonce value usually has a limited lifetime). Without proper credentials, the registrar will reject the de-registrations.

Given these concerns, we recommend that you consult with your Oracle systems engineer before adopting the use of forced unregistration.

SIP HNT Forced Unregistration Configuration

To enable SIP HNT forced unregistration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ```

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **options**—Set the options parameter by typing **options**, a Space, the option name **force-unregistration**, and then press Enter.

   ```
   ORACLE(sip-config)# options +force-unregistration
   ```

   If you type options force-unregistration, you will overwrite any previously configured options. In order to append the new option to the sip-config's options list, you must prepend the new option with a plus sign as shown in the previous example.

Adaptive HNT

This section explains how to configure adaptive HNT. The adaptive HNT expires feature allows the Oracle USM to automatically determine the maximum SIP REGISTER message expires time interval in order to keep each individual NAT pinhole open when performing SIP HNT. This feature applies only to SIP over UDP.

Overview

Without adaptive HNT, the Oracle USM keeps NAT pinholes open and port mapping cached by forcing the UAC to send frequent SIP REGISTER messages. It does so by setting the expires time to a short interval. Some NATs only need a message to be sent by the private client once every twenty minutes, while other NATs delete their cache/pinhole in thirty seconds if no messages appear. Given this large variation in time intervals, the Oracle USM's nat-interval (expire time) has been set to a low value in order to support as many NAT types as possible. However, CPU performance and scalability issues result from such a small refresh time, especially when there is a very large number of potential registered users.

When you use adaptive HNT, the Oracle USM waits for a time interval and then sends a SIP OPTIONS message to the UAC to see if it can still be reached. If the UAC can still be reached, the Oracle USM increases the timer and tries again. In case the pinhole closes because it has exceeded the NAT's cache time, the Oracle USM sets the expires time to be slightly longer than the time it tests using the OPTIONS method. This way, the UAC will send another REGISTER message shortly thereafter and impact on service will be minimal.

Adaptive HNT Example

An example call flow using adaptive HNT involves a basic HNT user and a Oracle USM. It begins when the Oracle USM receives and forwards the 200 OK for the REGISTER message. Then the Oracle USM sends an expires timer for slightly longer than the time for which to test; in this example, it begins the test for the amount of time set for the minimum NAT interval. It adds ten seconds to this time when it sends the expires timer. This way, there is time for the OPTIONS message to be sent before the REGISTER message is received (which would refresh the NAT's cache). The Oracle USM also tries to keep the REGISTER time short enough so that even if the NAT pinhole closes, there is minimal time before the UAC creates a new NAT binding by sending another REGISTER. Because a ten second interval may be too long, you might want to set this value to a better-suited time.

The test succeeds with a minimum test-timer because the UAC responded to the OPTIONS message. So the test-timer value is increased by thirty seconds and tried again. The expires time in the REGISTER message will be increased to the test-timer value plus ten seconds. This time, the UAC does not respond to the OPTIONS message even though it was sent multiple times. Because the OPTIONS fails, when the Oracle USM receives another REGISTER, it responds with the previously successful timer value (in this case, the minimum NAT interval).

However, if the OPTIONS request succeeds, then the Oracle USM persists with the test until it fails or until the maximum NAT timer value is reached. In this case, when the OPTIONS message fails, the Oracle USM uses the last successful test-timer value as the time for the expires header in the 200 OK for the REGISTER message.

Synchronize A-HNT Successful Timer to Standby

Adaptive HNT enables the Oracle USM to determine, through testing, an optimum SIP REGISTER expires time interval that keeps the NAT pinhole open. For an HA node, this successful time value is determined through testing by the active system and then replicated to the standby. If there is a switchover during the active system's testing process, then it will restart for that endpoint.

Adaptive NHT Configuration

You configure the SIP interface to set the state of this feature and to define the increments of time the Oracle USM uses to perform adaptive HNT. Remember that the Oracle USM uses the time you specify as the NAT interval, the supported time interval, as the basis on which to begin testing.

To configure adaptive HNT:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **sip-dynamic-hnt**—Enable this parameter if you want to use adaptive HNT. The default value is **disabled**. The valid values are:

   - enabled | disabled

5. **max-nat-interval**—Set the amount of time in seconds that testing should not exceed. The Oracle USM will keep the expires interval at this value. The default value is **3600**. The valid range is:

   - Minimum—0
   - Maximum—999999999

6. **nat-int-increment**—Set the amount of time in seconds to use as the increment in value in the SIP expires header. The default value is **10**. The valid range is:

   - Minimum—0
   - Maximum—999999999

7. **nat-test-increment**—Set the amount of time in seconds that will be added to the test timer. The default value is **30**. The valid range is:

   - Minimum—0
   - Maximum—999999999

SIP IP Address Hiding and NATing in XML

Adding to its topology hiding and NAT capabilities, the Oracle USM now performs those functions for pertinent IP addresses that are not part of the standard SIP message header format. Previously, such addresses were visible to the next hop in the SIP session path.

Note that this feature adds to the Oracle USM's pre-existing ability to perform this function for XML messages; this new support is specifically for the keyset-info message type.

For incoming SIP NOTIFY messages, the Oracle USM searches for the application/keyset-info +xml content type in the message. When it finds this content type, it searches further to detect the presence of <di:remote-uri> or <di:local-uri> XML tags and then NATs the IP addresses in the tags it finds. Specifically, the Oracle USM changes:

- The <di:remote-uri> IP address to be the egress SIP interface's IP address
- The <di:local-uri> IP address to be the Ip address of the next hop to which the message is being sent

Sample SIP NOTIFY with NATed XML

The following is a sample SIP NOTIFY message as it might arrive at the Oracle USM.

> **Note:**
>
> that it contains the <di:remote-uri> or <di:local-uri> XML tags on which the system will perform NAT; these lines appear in bold text.

```
NOTIFY sip:15615281021@10.152.128.253:5137;transport=udp SIP/2.0
To: 15615281021  <sip:15615281021@10.152.128.102:5080>;tag=5c93d019904036a
From: <sip:15615281021@10.152.128.102:5080>;tag=test_tag_0008347766
Call-ID: 3215a76a979d0c6
CSeq: 18 NOTIFY
Contact: <sip:15615281021@10.152.128.102:5080;maddr=10.152.128.102>
Via: SIP/2.0/UDP 10.152.128.102:5060;branch=z9hG4bK_brancha_0023415201
Event: keyset-info
Subscription-state: active;expires=2778
Accept: application/keyset-info+xml
Content-Type: application/keyset-info+xml
Content-Length: 599
Max-Forwards: 70
<?xml version="1.0"?>
<keyset-info xmlns="urn:ietf:params:xml:ns:keyset-info"
     version="16"
     entity="15615281021">
  <ki-data>
    <ki-state>"active"</ki-state>
    <ki-event>"unknown"</ki-event>
  </ki-data>
   <di:dialog id="dialog_id_201" call-
id="1395216611-1987932283256611-11-0884970552" local-tag="test_tag_0008347790"
direction="recipient">
    <di:state>trying</di:state>
    <di:duration>2778</di:duration>
    <di:local-uri>sip:15615281021@10.152.128.253:5137</di:local-uri>
    <di:remote-uri>sip:1004@10.152.128.102</di:remote-uri>
   </di:dialog>
</keyset-info>
```

Once the Oracle USM has completed the NAT process, the <di:remote-uri> and <di:local-uri> XML tags look like this

```
<di:local-uri>sip:15615281021@192.168.200.99:5137</di:local-uri>
<di:remote-uri>sip:1004@192.168.200.49</di:remote-uri>
```

because egress the SIP interface's IP address is 192.168.200.49 and the next hop's IP address is 192.168.200.99.

This feature does not require any configuration.

SIP Server Redundancy

This section explains how to configure SIP server redundancy. SIP server redundancy involves detecting that an upstream/downstream SIP signaling entity has failed, and adapting route policies dynamically to remove it as a potential destination.

Overview

You establish SIP server redundancy by creating session agents, which are virtual representations of the SIP signaling entities. These agents are then collected into a session agent group, which is a logical collection of two or more session agents that behaves as a single aggregate entity.

Rather than direct signaling messages to a single session agent (IP), the signaling message is directed to a session agent group (SAG). The group will have a set distribution pattern: hunt, round robin, proportionally distributed, and so on. Signaling is spread amongst the agents using this chosen pattern.

You direct the signaling message by configuring a route policy, known as a local policy, which determines where SIP REQUESTS should be routed and/or forwarded. The values in the To and From headers in the SIP REQUEST are matched with the content of the local policy within the constraints set by the session agent's previous hop value and SIP interface values such as the list of carriers.

To summarize, you need:

- two or more session agents
- a session group containg those session agents
- a local policy which directs traffic to the session agent group

Configuration Overview

You make a session agent group a target by using a local policy to select the next hop from the members of a session agent group. You need to set the replace URI field of the configured local policy to enabled; which causes NAT rules such as realm prefixing to be overridden. The replace URI field allows you to indicate whether the local policy's value is used to replace the Request-URI in outgoing requests. This boolean field can be set to either enabled or disabled.

When the SIP NAT's route home proxy field is set to forced, it forces the Request to be forwarded to the home proxy without using a local policy. When this option is set to either disabled or enabled and the Request-URI matches the external address of the SIP NAT, the local policy is used.

However, the local policy only replaces the Request-URI when the original Request-URI matches the SBC's IP address or hostname. This behavior is in accordance with that described in RFC 3261. The original Request-URI will be the home proxy address value (the home address of the SIP NAT into the backbone) and not the Oracle USM (USM) address.

Using strict routing, the Request-URI would be the next hop, but the message would also include a Route header with the original Request-URI. With loose routing, the Request-URI remains unchanged and the next hop value is added as the top Route header.

Sometimes the next hop field value must replace the Request-URI in the outgoing request, even if the original Request-URI is not the USMC. To accomplish this, an option has been added to

the local policy that causes the next hop value to be used as the Request-URI and prevents the addition of Route headers. This option is the replace uri value in the local policy.

The following table lists the policy attributes for the local policy:

Parameter	Description
next hop	IP address of your internal SIP proxy. This value corresponds to the IP address of the network interface associated with the SIP proxy.
realm	Number of the port associated with the SIP port.
replace uri	Stores the transport protocol used for sending an receiving signaling messages associated with the SIP port.
allow anonymous	Indicates whether this SIP port allows anonymous connections from session agents.

> **Note:**
>
> You should also define the ping method intervals for the session agents so that the USM can detect when the agents are back in service after failure.

SIP Server Redundancy Configuration

To enable replace URI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **local-policy** and press Enter to access the system-level configuration elements. The system prompt changes.

   ```
   ORACLE(configure)# local-policy
   ORACLE(local-policy)#
   ```

3. Type **policy-attributes** and press Enter. The system prompt changes.

   ```
   ORACLE(local-policy)# policy-attributes
   ORACLE(local-policy-attributes)#
   ```

   From this point, you can configure policy attributes for the local policy. To see all local policy attribute options, enter a **?** at the system prompt.

4. **action**—Set this parameter to **replace-uri**, which causes NAT rules such as realm prefixing to be overridden. The default value is **none**. Valid values are:

   - none | replace-uri | redirect

     The replace URI field allows you to indicate whether the local policy's value is used to replace the Request-URI in outgoing requests. This boolean field can be set to either enabled or disabled.

Administratively Disabling a SIP Registrar

The Oracle USM's registration cache feature is commonly used to support authorization. It also allows the Oracle USM to respond directly to SIP REGISTER requests from endpoints rather than forwarding every REGISTER message to the Registrar(s). In the Oracle USM, Registrars

are frequently configured as session agents, and an association between each endpoint and its Registrar is stored with the registration cache information.

In Release 4.0.1 and later, the i**nvalidate-registrations** parameter in the session agent configuration enables the Oracle USM to detect failed Registrar session agents and automatically forward subsequent REGISTER requests from endpoints to a new Registrar. You can now perform the same behavior manually through a new ACLI command. When you use this command, the Oracle USM acts as though the registrations have expired.

For each SIP session agent, you can enable the manual trigger command, and then use the command from the main Superuser ACLI prompt. The **reset session-agent** command provides a way for you to send a session agent offline. Session agents can come back online once they send 200 OK messages the Oracle USM receives successfully.

Without using the manual trigger, session agents can go offline because of they do not respond to pings or because of excessive transaction timeouts. However, you might not want to use these more dynamic methods of taking session agents out of service (and subsequently invalidating any associated registrations). You can disable both of these mechanisms by setting the following parameters to 0:

- **ping-interval**—Frequency (amount of time in seconds) with which the Oracle USM pings the entity the session agent represents)
- **ttr-no-response**—Dictates when the SA (Session Agent) should be put back in service after the SA is taken OOS (Out Of Service) because it did not respond to the Oracle USM

However, you can still use the new SIP manual trigger even with these dynamic methods enabled; the trigger simply overrides the configuration to send the session agent offline.

Considerations for Implicit Service Route Use

When implicit service route support is enabled for a SIP interface (in IMS applications), the Oracle USM stores the Service Route URIs from the Service-Route headers that are included in 200 OK responses to REGISTER messages. Subsequently, and even when a session agent is rendered invalid, re-REGISTER messages follow the route stored in the cache instead of using the one defined in the Oracle USM.

However, you might not want to use this behavior when you send session agents offline. If you instead want use the route defined in the Oracle USM, then you need to configure the SIP interface option called **route-register-no-service-route**.

Manual Trigger Configuration

This section shows you how to enable the manual trigger for sending session agents out of service, and how to then use the trigger from the command line. This section also shows you how to verify that you have successfully put a session agent out of service.

To enable a SIP session agent to manually trigger it to go out of service:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **invalidate-registrations**—Set this parameter to enabled if you want to use the manual trigger to send this session agent offline (and therefore invalidate the registrations associated with it). The default is disabled.

5. Save and activate your configuration.

   To use the manual trigger that sends session agents offline:

6. Note the hostname value (typically the IP address of the endpoint) for the session agent you want to put out of service. You use this name as an argument in the ACLI command to use the manual trigger.

7. At the Superuser prompt, type **reset session-agent**, a Space, and the hostname value for the session agent. The press Enter.

   ```
   ORACLE# reset session-agent 192.168.20.45
   ```

   If you enter a session agent that does not exist, the system notifies you that it cannot carry out the reset.

Manual Trigger Confirmation

To confirm that a session agent has been sent offline:

1. Use the **show sipd endpoint-ip** command to confirm the session agent state.

   ```
   ACMEPACKET# show sipd endpoint-ip 1016
   User <sip:1016@172.18.1.80>
           Contact exp=3582
               UA-Contact: <sip:1016@192.168.1.132:5060> UDP
                   realm=access local=172.18.1.132:5060
   UA=192.168.1.132:5060
       SD-Contact: <sip:1016-o3badgbbnjcq5@172.18.2.80:5060> realm=core
       Call-ID: 1-7944@192.168.1.132'
       SR=172.18.2.92
       SA=172.18.2.93
       Service-Route='<sip:test@s-cscf::5060;orig;lr>'
   ACMEPACKET# reset session-agent 172.18.2.92
   Accepted
   Reset SA failover timer
   ACMEPACKET# show sipd endpoint-ip 1016
   User <sip:1016@172.18.1.80>
           Contact <invalidated> exp=3572
               UA-Contact: <sip:1016@192.168.1.132:5060> UDP
                   realm=access local=172.18.1.80:5060
   UA=192.168.1.132:5060
       SD-Contact: <sip:1016-o3badgbbnjcq5@172.18.2.80:5060> realm=core
       Call-ID: 1-7944@192.168.1.132'
       SR=172.18.2.92 (failed 2 seconds ago)
       SA=172.18.2.93
       Service-Route='<sip:test@s-cscf::5060;orig;lr>'
   ACMEPACKET#
   ```

In the above ACLI example the first iteration of the **show sip endpoint-ip** command provides information for the in-service 172.18.2.92 session agent; the second command iteration displays information for the now out-of-service session agent.

Surrogate Agent Refresh on Invalidate

Surrogate agent registrations normally only re-register when nearing their expiration time. When a registrar fails, the surrogate agent will wait until the expiration time to refresh the registration with an in-service registrar.

You can configure your Oracle USM to immediately refresh the surrogate agent registrar with an in-service registrar by enabling the existing parameter **invalidate-registrations**.

Invalidate Registrations

An existing feature called **invalidate-registrations** located in the session agent keeps track of when surrogate agents go out of service. When REGISTER messages are received, registration entries that had out-of-service session agents since the last REGISTER will always allow the message through to the registrar (as opposed to responding directly from the cache).

The **invalidate-registrations** parameter in session agent configuration enables the Oracle USM to detect failed Registrar session agents.

If invalidate-registrations is enabled for the session agent, a response from a surrogate REGISTER that contains a service-route header that corresponds to a session-agent is installed to the registration cache entry.

The surrogate-agents are scanned. Surrogate agents with registration entries matching the out-of-service registrar have their timer reset to initiate a refresh. For an immediate refresh, the registration entry will only be considered when the service-route session agent goes out-of-service. The service-route session agent takes precedence and any previous registrar session agent w ill not be considered for an immediate refresh of the surrogate-agent registration.

Performance Impact

In cases with a large number of surrogate-agent registrations, there may be an impact to CPU usage when a session-agent goes out-of-service. All of the surrogate-agent registrations are scanned at that time. Refresh registrations are then sent out on timers.

Media Inactivity Timer Configuration

To disable the media inactivity timer for calls placed on hold:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration you want to edit.

4. **invalidate-registration**—Set this parameter to **enabled** if you want to use the manual trigger to send this session agent offline (and therefore invalidate the registrations associated with it). The default is **disabled**.
5. Save and activate your configuration.

Support for Encoded Multipart Message Bodies

SIP messages and responses may arrive at the Oracle USM with encoded multipart message bodies, such that the content of the body is unreadable. This information may be encoded for the purpose of compressing the data. Normally, the Oracle USM would consider the body invalid and reject the entire message, replying to the sender with a `400 Invalid Body` error response. The user, however, can configure the **sip-config** option, **proxy-content-type-encodings**, allowing the Oracle USM to accept, process and forward messages containing these encoded parts. This configuration causes the Oracle USM to ignore the encoding, identify the end of the message via content length, and pass the message towards its intended recipient with the multipart body fully encoded.

The condition that triggers this functionality is the Oracle USM recognizing the presence of a multipart message body and the definition of the encoding type within the message.

```
NOTIFY sip:user@example.com SIP/2.0
Via: SIP/2.0/TCP
...
Content-Type: multipart/mixed;boundary="imdn-boundary"
Content-Encoding: gzip

... Encoded multipart content ...
```

When configured, the **proxy-content-type-encodings** is simply a list of strings. The Oracle USM looks to match the string defining the content encoding with a string in the list to proceed with the functionality.

The Oracle USM only performs this procedure on messages encoded with types for which it is configured and that are properly formed. Examples of when the Oracle USM does not perform this procedure include:

- The Oracle USM receives a message with an encoded multipart message block, but **proxy-content-type-encodings** list is empty. In this case, the Oracle USM responds with a `415 Unsupported Media Type`.
- The message arrives with a multipart message body with encoding for which the **proxy-content-type-encodings** is configured, but there is no terminating boundary. In this case, the Oracle USM replies with a `400 Invalid Body error`.
- The Oracle USM receives a response with an encoded multipart message block, but **proxy-content-type-encodings** list is empty. In this case, the Oracle USM simple drops the response.

Multipart Message Encoding Support Configuration

The procedure below provides the steps needed to configure the Oracle USM for multipart message encoding support.

To have your Oracle USM proxy messages despite the presence of the specified multipart message encoding:

1. In Superuser mode, use the following command sequence to access the **sip-config** element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure encoding support.

2. Configure the desired encoding types using the option followed by comma-separated coding types.

   The example below configures support for gzip and compressed encoding.

   ```
   ORACLE(sip-config)# options +proxy-content-type-encodings=gzip,compress
   ```

3. Type **done** and exit configuration mode. Save and activate your configuration.

SIP Distributed Media Release

This section explains how to configure distributed media release (DMR). SIP DMR lets you choose whether to include multi-system (multiple Oracle USMs) media release information in SIP signaling requests sent into a specific realm.

Overview

The SIP DMR feature lets RTP/RTCP media be sent directly between SIP endpoints (for example, SIP phones or user agents) without going through a Oracle USM; even if the SIP signaling messages traverse multiple Oracle USMs. It encodes IPv4 address and port information for the media streams described by the media, for example SDP.

With SIP DMR, the media realm and IPv4 address and port information from the UA's SDP is encoded into SIP messages (either in the SIP header or in the SDP) as they enter the backbone network. The information is decoded by a Oracle USM from SIP messages that come from the backbone network. The decoded address and port information is put into the SDP sent the UAs in the access (private/customer) network.

This functionality lets the RTP/RTCP flow directly between the UAs in the access network without traversing the Oracle USMs and without passing into the backbone network. The media can then flow directly between the two SIP endpoints in the same network, if it is serviced by multiple Oracle USMs.

You can enable this feature on a per-realm basis and multiple realms can be supported.

Endpoint Locations

You can configure the Oracle USM to release media when the source and destination of the call are in the same network, customer VPN, or customer LAN. In architectures that use DMR, the

Oracle USM is only part of the media path for traffic that originates and terminates in different networks.

If configured to do so, the Oracle USM can release media:

- Between endpoints supported by a single Oracle USM
  In the same network/VPN

  In the same network behind the same NAT/firewall
- Between endpoints supported by multiple distributed Oracle USMs
  In the same network/VPN

Location of the Encoded Information

Encoded media release information can appear in three different places:

- SDP attribute
  Media release data can be encoded into an SDP attribute in the SIP message body (for example, media-release=sdp;acme-media). The encoded data is placed into an acme-media attribute in the SDP:

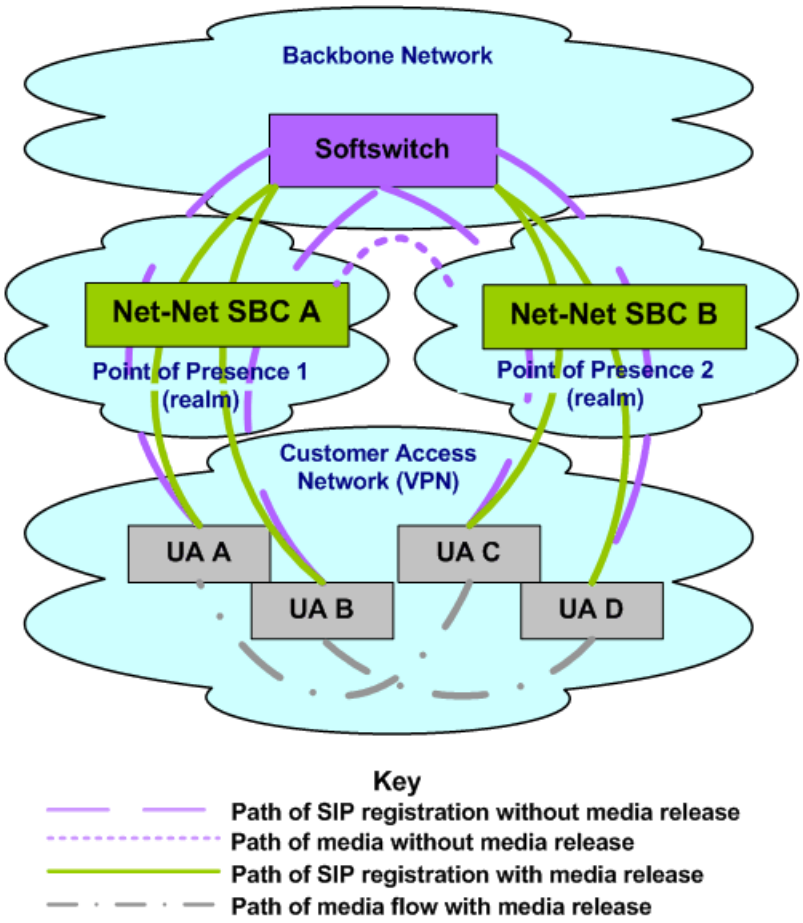
  ```
  a=acme-media:<encoded-media-interface-info>
  ```
- SIP header parameter
  Media release data can be placed in a header parameter of a SIP header (for example, media-release=Contact;acme-media). The encoded data is placed into an acme-media parameter in the Contact header:

  ```
  Contact: <sip:1234@abc.com>;acme-media=<encoded-media-interface-info>
  ```
- SIP header
  Media release data can appear in a SIP header (for example, media-release=P-Media-Release). The encoded data is placed into a P-Media-Release header:

  ```
  P-Media-Release: <encoded-media-interface-info>
  ```

Example Distributed Media Release

The following example shows the network diagram for DMR in a multiple-site VPN environment supported by multiple, distributed Oracle USMs.



As shown in the network diagram, UA A and UA B register with the softswitch through Oracle USM A while UA C and UA D register with the softswitch through Oracle USM B. Without DMR, the media for calls between UA A/UA B and UA C/UA D is steered through both Oracle USM A and Oracle USM B.

With SIP DMR, the media realm and IPv4 address and port information from the UA's Session Description Protocol (SDP) is encoded into SIP messages (either in the SIP header or in the SDP) as they enter the backbone (public/service provider) network. The information is decoded from SIP messages that come from the backbone network. The decoded address and port information is put into the SDP sent to the UAs in the access (private/customer) network. This functionality allows for the RTP/RTCP to flow directly between the UAs in the access network without traversing the Oracle USMs and without passing into the backbone network.

Overview of SIP DMR Configuration

To configure SIP DMR:

1. Edit the SIP config element's option field.

   The media-release="<header-name>[;<header-param>]" option defines how the SIP distributed media release feature encodes IPv4 address and port information. If the media-release parameter is configured in the options field but no header is specified, the parameter value of P=Media-Release will be used. This parameter is optional and is not configured by default.

2. Enable SIP DMR for the entire realm by setting the realm config element's msm release field to enabled.

The media IPv4 address and port information is encoded into outgoing SIP messages and decoded from incoming SIP messages for all of the realms (in each realm-config element) with which the SIP distributed media release will be used.

> **Note:**
>
> You can also use the realm config element's mm in network field to release the media back to a connected network that has multiple realms. This field is not specific SIP distributed media release and it is not required for the SIP DMR to work. However, if this field is set to enabled and the ingress and egress realms are part of the same network interface, it lets the Oracle USM release the media.

SIP DMR Configuration

To configure media release:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.
5. After the Space, type the media release information in the following format:

   ```
   media-release="<header-name>[;<header-param>]"
   ```

   - header-name either refers to the SIP header in which to put the information or to the special header-name value of sdp to indicate the information should be put into the SDP.
   - parameter-name refers to the header parameter name in which to put the information or, in the case of the special header-name value of sdp, to the SDP attribute name in which to put the information.

     For example:

     ```
     ORACLE(sip-config)# options media-release=P-Media-Release
     ```

6. Press Enter.

> **Note:**
>
> If the media-release parameter is configured in the options field, but no header is specified, then the parameter value of P-Media-Release will be used. P-Media-Release is a proprietary header and means that the media will be encoded in the SIP header with this name.

The following example shows where the encoded information (for example, SDP data) is passed.

```
media-release="P-Media-Release"
media-release="Contact;acme-media"
media-release="sdp;acme-media"
```

Configuring the Realm

You need to set the each realm config element's msm release field to enabled for all the realms for which you want to use SIP DMR.

Although the mm in network field is not specific to the SIP distributed media release feature, it can be used to release the media back to a connected network that has multiple realms. This field does not need to be configured in order for the SIP distributed media release feature to work. However, if this field is set to enabled and the ingress and egress realms are part of the same network interface, it lets the Oracle USM release the media.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **msm-release**—Enable DMR within this realm on this Oracle USM. The default value is **disabled**. The valid values are:

   - enabled | disabled

5. Repeat for each realm on which you want to enable DMR.

Add-On Conferencing

This section explains how to configure the add-on conferencing functionality. It also includes a description of the SIP B2BUA functionality related to the SIP add-on conferencing. This description includes information about Contact header mapping and processing and Refer-to header processing.

Overview

SIP add-on conferencing lets you:

- Use the Oracle USM's add-on conferencing feature for network architectures in which the conference initiator is located on a different network than that of the media server.
- Configure the Oracle USM to enable Contact header mapping for the Refer-To header.

Caveats

The following caveats are associated with add-on conferencing:

- Contact header mapping is not replicated on the standby Oracle USM in an HA Oracle USM pair architecture.
- Upon switchover, any conferences in progress remain in progress, but no new parties can be invited to or join the conference.
- By default, the Oracle USM does not map SIP Contact headers for reasons of performance.

Add-On Conferencing Scenario

The add-on conferencing scenario described in the following example applies to a network architecture involving the Oracle USM and a media server that is located on a different network from the other conference participants. In this scenario, the Oracle USM resides on a standalone network that connects two additional, separate networks.

Some network architectures have a media server on a different network from the one on which the phones reside. In this scenario, all requests and/or responses going from the phones (Phone A, Phone B, or Phone C) to Media Server D and vice versa are translated according to their corresponding SIP-NAT. All headers subjected to NAT are encoded and decoded properly as they traverse the Oracle USM, except for the Contact header. This exception occurs because the SIP process on the Oracle USM runs as a SIP B2BUA and not as a SIP proxy.

The SIP B2BUA re-originates the Contact headers of the User Agents (UAs) participating in SIP sessions with local Contact headers to make sure that they receive all future in-dialog requests. For an in-dialog request, the B2BUA can identify the dialog and find the Contact URI of the other leg of the call.

The Oracle USM add-on conferencing feature applies to situations when the Contact URI is used in another dialog. In such a case, the SIP B2BUA will not be able to find the correct dialog that retrieves the correct Contact URI of the other leg if it needs to replace the Contact URI.

Using the SIP add-on conferencing, the SIP B2BUA on the Oracle USM can map the Contact headers it receives to the Contact headers it creates. It can also convert the Refer-To URI to the correct value required for forwarding the REFER request.

SIP B2BUA Functionality

This section describes the role of the Oracle USM's SIP B2BUA in the add-on conferencing scenario that requires Contact header mapping for the Refer-To header.

When the Oracle USM starts up, the SIP B2BUA reads and parses the list of options in the SIP configuration. If the refer to uri prefix is an appropriate value (it is not an empty string), the

Oracle USM will have a text prefix value the media server can use to denote a conference ID in its Contact header. With this information, the SIP B2BUA sets up a Contact header mapping.

You configure the Oracle USM to enable Contact header mapping for the Refer-To header by editing the SIP config options parameter. The SIP B2BUA on the Oracle USM can then map the Contact headers it receives to the Contact headers it creates.

Contact Header Processing

The Contact header mapping matches a Contact header that contains the refer to URI prefix to the corresponding Contact header that the Oracle USM's SIP B2BUA re-originates. Contact headers that do not contain the refer to URI prefix are not mapped (so that performance of the Oracle USM is minimally affected).

Only the Contact header in an INVITE request and its 200 OK response are checked for the refer to URI prefix and added to the Contact header mapping. Contact headers appearing in other SIP requests/responses are not checked.

Target Mapping and Conferences

If the Oracle USM is configured to enable Contact header mapping for the Refer-To header, then Contact header target maps are established for each individual call. The Oracle USM's SIP B2BUA uses these maps to allow the media server to connect the conference initiator with the conferenced-in parties.

Prior to terminating the call (hanging up), the conference initiator can contact other parties and invite those additional parties to join the conference. These other parties can join the existing conference because the target mapping for the conference is still in effect on the Oracle USM.

Once the conference initiator hangs up, the Oracle USM discards the mapping from the conference.

Refer-To Header Processing

When a Refer-To header is present in a REFER request that arrives at the SIP B2BUA after the incoming request is properly translated according to its SIP-NAT, the SIP B2BUA follows these steps:

1. The SIP B2BUA parses the Refer-To URI.
2. If the user part of the Refer-To URI contains the refer to URI prefix, the SIP B2BUA searches the Contact header mapping for a match of the user part of the URI.

   If the user part of the Refer-To URI does not contain the refer to URI prefix, the SIP B2BUA leaves the existing Refer-To URI unchanged.
3. If the user part of the Refer-To URI contains the refer to URI prefix and a match of the Refer-To URI is found, the SIP B2BUA replaces the existing Refer-To URI with the URI of the corresponding Contact URI stored in the matched record. This replacement enables the NAT function to properly decode the replacement URI and change it back to the form originally received by the Oracle USM. As a result, the correct conference ID is restored in the Refer-To header prior to the request being sent to its next hop.

   If the user part of the Refer-To URI contains the refer to URI prefix but a matched URI cannot be found, the SIP B2BUA will leave the existing Refer-To URI unchanged and will write a WARNING level log message to record the failure.

Add-on Conferencing Configuration

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.

5. After the Space, type the add-on conferencing information in the following format:

   ```
   options refer-to-uri-prefix="conf="
   ```

   For example:

   ```
   ORACLE(sip-config)# options refer-to-uri-prefix="conf"
   ```

6. Press Enter.

SIP REFER Method Call Transfer

In prior releases, the Oracle USM supports the SIP REFER method by proxying it to the other UA in the dialog. A handling mode has been developed for the REFER method so that the Oracle USM automatically converts a received REFER method into an INVITE method, thus allowing the Oracle USM to transfer a call without having to proxy the REFER back to the other UA.

This function can be configured for a specified SIP interface, a realm, or a session agent. When all three elements have the SIP REFER method call transfer functionality configured, the session-agent configuration takes precedence over realm-config and sip-interface configurations. If session-agent is not configured, and realm-config and sip-interface are, realm-config takes precedence.

The Oracle USM has a configuration parameter giving it the ability to provision the handling of REFER methods as call transfers. The parameter is called refer-call-transfer. When this feature is enabled, the Oracle USM creates an INVITE message whenever it receives a REFER. The Oracle USM sends this INVITE message to the address in the Refer-To header. Included in the INVITE message is all the unmodified information contained in the REFER message. The previously negotiated codec is also still used in the new INVITE message. NOTIFY and BYE messages are sent to the UA upon call transfer completion.

If a REFER method is received containing no Referred-By header, the Oracle USM adds one, allowing the Oracle USM to support all call agent screen applications.

In addition, the SIP REFER method call transfer feature supports the following:

- Both unattended and attended call transfers

- Both successful and unsuccessful call transfers
- Early media from the Referred-To party to the transforee
- REFER method transfer from different sources within the destination realm
- The REFER event package as defined in RFC 3515. This applies for situations where multiple REFER methods are used within a single dialog.
- Third party initiated REFER method signalling the transfer of a call by associating the REFER method to the dialogue via the REFER TargetDialog.
- The Referred-To party can be both in a different realm (and thus a different steering pool) from the referrer, and in the same realm
- The associated latching should not prohibit the Referred-To party from being latched to while the referee is still sending media.

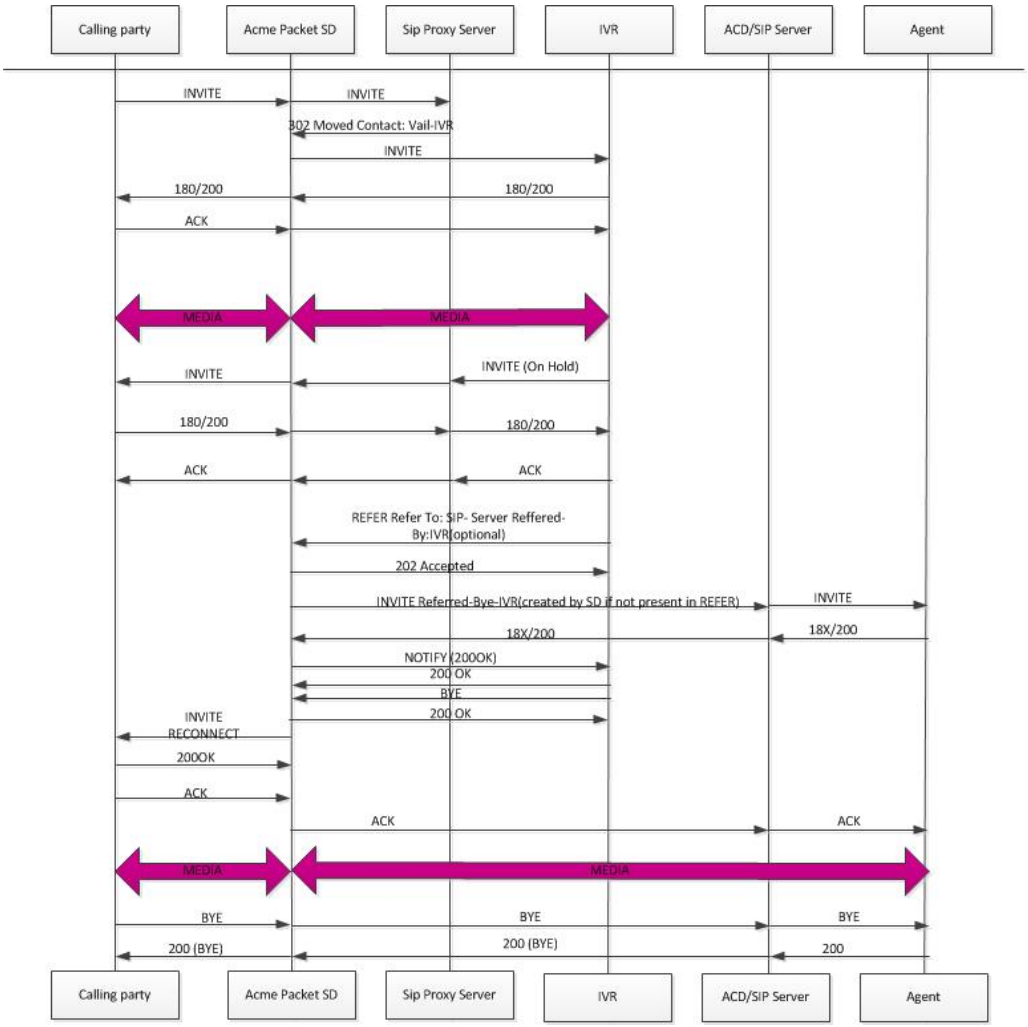
Unsuccessful Transfer Scenarios

The Oracle USM does not successfully handle the following failed, unusual, and unexpected transfer scenarios:

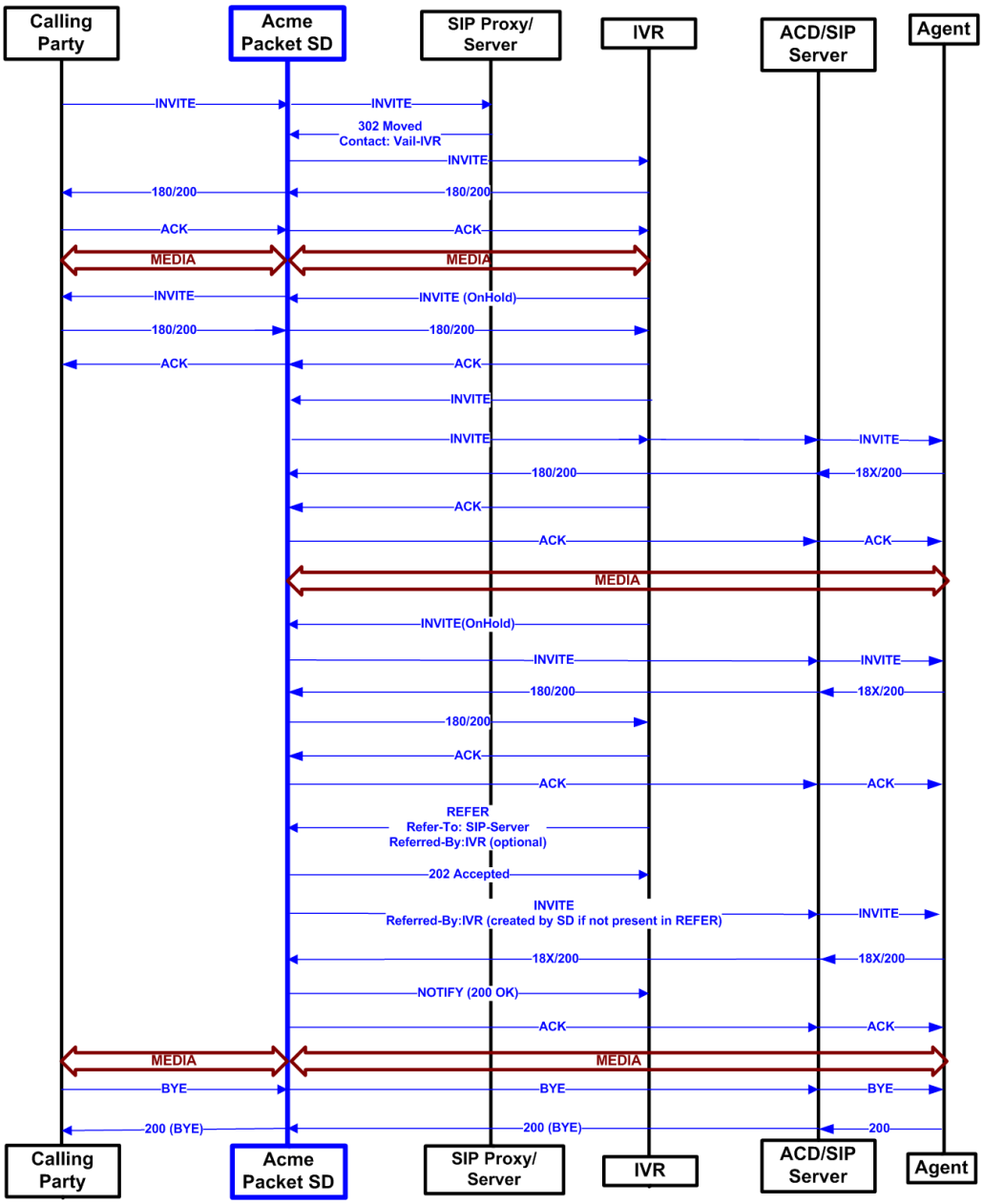
- The new INVITE to the Referred-To party gets challenged, the Oracle USM does not answer the challenge. It is treated with the 401/407 response just as any other unsuccessful final response.
- The header of the REFER message contains a method other than INVITE or contains URI-parameters or embedded headers not supported by the Oracle USM.
- The Oracle USM shall allow the Referred-To URI that happens to resolve to the same next-hop as the original INVITE went to, to do so.
- The Oracle USM ignores any MIME attachment(s) within a REFER method.
- The Oracle USM recurses (when configured to do so) when the new INVITE sent to the Referred-To party receives a 3xx response.
- The transferee indicated support for 100rel, and the original two parties agreed on using it, yet the Referred-To party does not support it.
- The original parties negotiated SRTP keys.
- The original parties agreed on a codec using a dynamic payload type, and the Referred-To party happens to use a different dynamic payload number for that codec.

Call Flows

The following is an example call flow for an unattended call transfer:



The following is an example call flow of an attended call transfer:



SIP REFER Method Configuration

To enable SIP REFER method call transfer in the realm-config:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
5. Save and activate your configuration.

   To enable SIP REFER method call transfer in the sip-interface:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

8. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-config)#
   ```

9. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
10. Save and activate your configuration.

    To enable SIP REFER method call transfer in a realm:

11. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

12. Type **media-manager** and press Enter.

    ```
    ORACLE(configure)# media-manager
    ORACLE(media-manager)#
    ```

13. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

    ```
    ORACLE(media-manager)# realm-config
    ORACLE(realm-config)#
    ```

14. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
15. Save and activate your configuration.

    To enable SIP REFER method call transfer in the session-agent:

16. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

17. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```

18. Type **session-agent** and press Enter.

    ```
    ORACLE(media-manager)# session-agent
    ORACLE(session-agent)#
    ```

19. **refer-call-transfer**—Set to **enabled** to enable the refer call transfer feature. The default for this parameter is **disabled**.
20. Save and activate your configuration.

REFER-Initiated Call Transfer

In prior releases, the Oracle USM supported REFER-initiated call transfer either by proxying the REFER to the other User Agent in the dialog, or by terminating the received REFER and issuing a new INVITE to the referred party. These static alternate operational modes could be configured for specific SIP interfaces, realms, or session agents.

Release S-C6.2.0 enhances support with an additional operational mode that determines on a call-by-call basis whether to proxy the REFER to the next hop, or terminate the REFER and issue an INVITE in its stead.

> **Note:**
>
> With the release of Version S-C6.2.0, support for REFER-initiated call transfer is no longer available for SIP interfaces; support must be configured for realms and/or session agents.

Version S-C6.2.0 provides a new configuration parameter **dyn-refer-term**, and a revised **refer-call-transfer** parameter (both available in realm-config configuration mode) that specify call transfer modes.

With the **refer-call-transfer** parameter set to **disabled** (the default), all received REFERs are simply proxied to the peer User Agent.

With the **refer-call-transfer** parameter set to **enabled**, the Oracle USM terminates all REFERs, generates a new INVITE, and sends the INVITE to the address in the Refer-To header.

With the **refer-call-transfer** parameter set to **dynamic** (a new value introduced with Version S-C6.2.0), the Oracle USM determines REFER handling on a call-by-call basis as follows:

1. Check the **refer-call-transfer** value for the session agent from which the REFER was received, or for ingress realm (the realm that received the REFER).

   If the value is disabled, proxy the REFER to the peer User Agent, to complete REFER processing.

   If the value is enabled, terminate the REFER and issue an new INVITE to the referred party, to complete REFER processing.

   If the value is dynamic, identify the next hop egress realm.
2. Check the **dyn-refer-term** value for the next hop egress realm.

   If the **dyn-refer-term** value is disabled (the default), proxy the REFER to the next hop to complete REFER processing.

   If the **dyn-refer-term** value is enabled, terminate the REFER and issue an new INVITE to the referred party to complete REFER processing.

Supported Scenarios

In the basic scenario for REFER initiated call transfer, a call is established between two User Agents (Alice and Bob). User Agent Bob then sends a REFER request to transfer the call to a third User Agent Eva. With dynamic call-transfer enabled, the Oracle USM (USM) prevents the REFER from being sent to Alice and generates the INVITE to Eva.

If the INVITE to Eva succeeds, the USM sends a re-INVITE to Alice modifying the SIP session as described in Section 14 of RFC 3261, *SIP: Session Initiation Protocol*. At this point the USM cancels the original dialog between the USM and Bob.

If the INVITE to Eva fails, call disposition depends on whether or not Bob issued a BYE after the REFER call transfer. If the Oracle USM did receive a BYE from Bob (for instance, a blind transfer), it proxies the BYE to A. Otherwise, the USM retains the original SIP session and media session, thus allowing Bob to re-establish the call with Alice by sending a re-INVITE. In this case, the USM sets a timer (32 seconds), after which a BYE will be sent.

If a REFER method is received containing no Referred-By header, the USM adds one, allowing the USM to support all call agent screen applications.

In addition, the SIP REFER method call transfer feature supports the following:

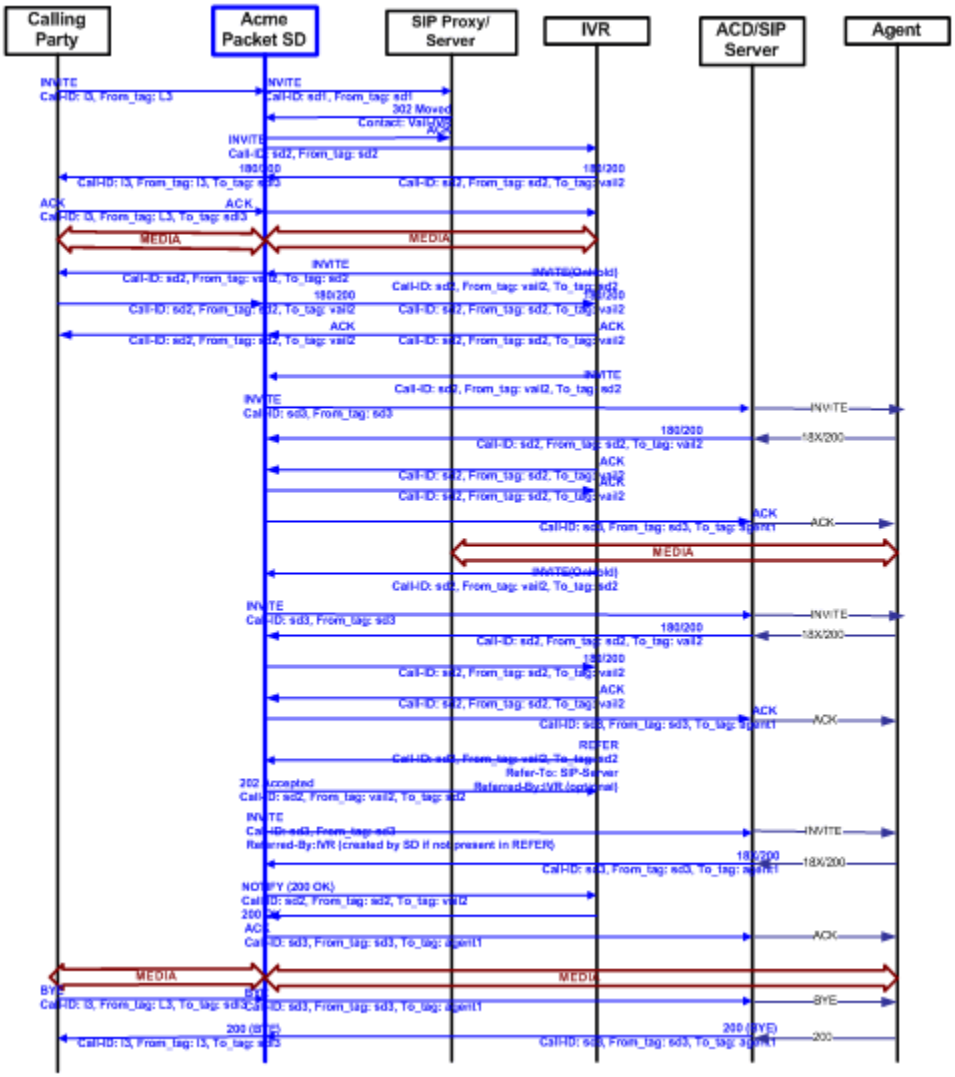
- Both unattended and attended call transfers
- Both successful and unsuccessful call transfers
- Early media from the Referred-To party to the transferee
- REFER method transfer from different sources within the destination realm
- The REFER event package as defined in RFC 3515. This applies for situations where multiple REFER methods are used within a single dialog.
- Third party initiated REFER method signalling the transfer of a call by associating the REFER method to the dialogue via the REFER TargetDialog.
- The Referred-To party can be both in a different realm (and thus a different steering pool) from the referrer, and in the same realm
- The associated latching should not prohibit the Referred-To party from being latched to while the referee is still sending media.

The USM does not successfully handle the following anomalous transfer scenarios:

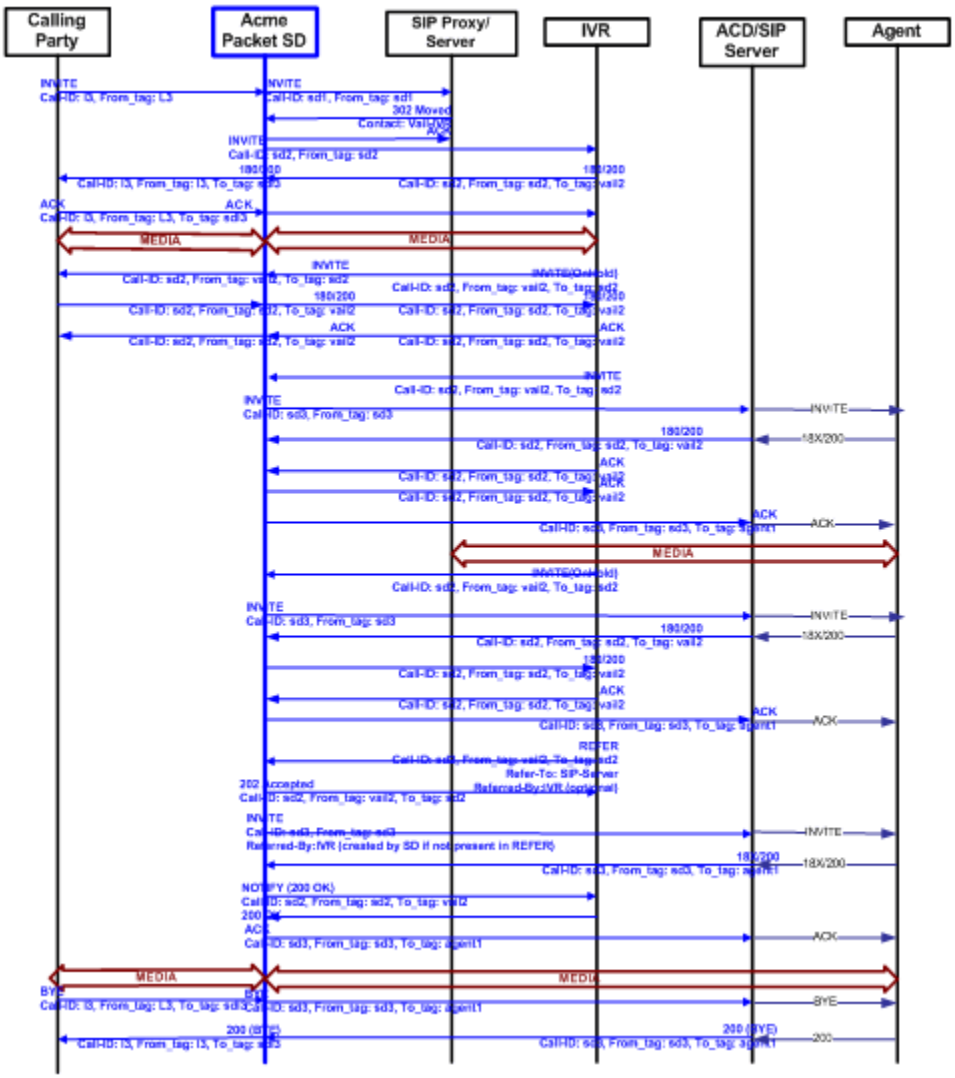
- The new INVITE to the Referred-To party gets challenged — the USM does not answer the challenge. It is treated with the 401/407 response just as any other unsuccessful final response.
- The header of the REFER message contains a method other than INVITE or contains URI-parameters or embedded headers not supported by the USM.
- The USM shall allow the Referred-To URI that happens to resolve to the same next-hop as the original INVITE went to, to do so.
- The USM ignores any MIME attachment(s) within a REFER method.
- The USM recurses (when configured to do so) when the new INVITE sent to the Referred-To party receives a 3xx response.
- The transferee indicated support for 100rel, and the original two parties agreed on using it, yet the Referred-To party does not support it.
- The original parties negotiated SRTP keys.

Call Flows

The following is an example call flow for an unattended call transfer:



The following is an example call flow of an attended call transfer:



REFER Source Routing

If, after the conclusion of static or dynamic REFER handling, the REFER is terminated and a new INVITE issued, users now can specify a policy lookup behavior based upon either the source realm of the calling party (the INVITE originator), or the source realm of the referring party (the REFER originator).

Behavior is controlled by a new **refer-src-routing** parameter in the **sip-config** configuration element.

disabled, the default value, specifies that the Oracle USM performs a policy lookup based on the source realm of the calling party.

enabled specifies that the Oracle USM performs a policy lookup based on the source realm of the referring party.

REFER Source Routing Configuration

To enable realm-based REFER method call transfer:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **refer-call-transfer** — Retain the default (**disabled**) to proxy all REFERs to the next hop. Use **enabled** to terminate all REFERs and issue a new INVITE. Use **dynamic** to specify REFER handling on a call-by-call basis, as determined by the value of the **dyn-refer-term** parameter.

5. **dyn-refer-term** (meaningful only when **refer-call-transfer** is set to dynamic) — Retain the default (**disabled**) to terminate the REFER and issue a new INVITE. Use **enabled** to proxy the REFER to the next hop.

6. Save and activate your configuration.

   To enable session-agent-based REFER method call transfer:

7. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

8. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

9. Type **session-agent** and press Enter.

   ```
   ORACLE(media-manager)# session-agent
   ORACLE(session-agent)#
   ```

10. **refer-call-transfer** — Retain the default (**disabled**) to proxy all REFERs to the next hop. Use **enabled** to terminate all REFERs and issue a new INVITE. Use **dynamic** to specify REFER handling on a call-by-call basis, as determined by the value of the **dyn-refer-term** parameter.

11. Save and activate your configuration.

    To specify policy lookup for a newly generated INVITE:

12. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

13. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```

14. Type **sip-config** and press Enter.

    ```
    ORACLE(configure)# sip-config
    ORACLE(sip-config)#
    ```

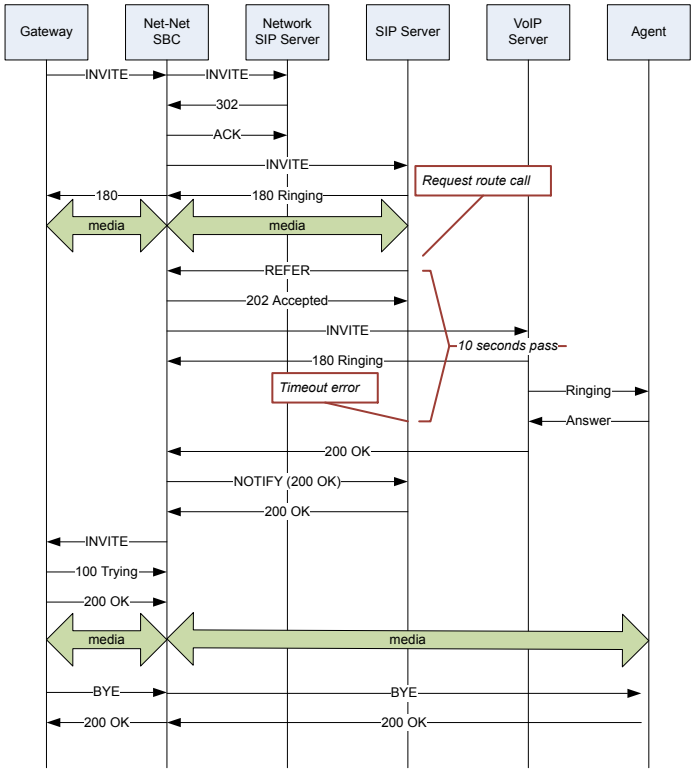
15. **refer-src-routing** — Retain the default (**disabled**) to perform a policy lookup based upon the source realm of the calling party (the issuer of the original INVITE). Use enabled to perform a policy lookup based upon the source realm of the referring party (the issuer of the REFER).
16. Save and activate your configuration.

180 & 100 NOTIFY in REFER Call Transfers

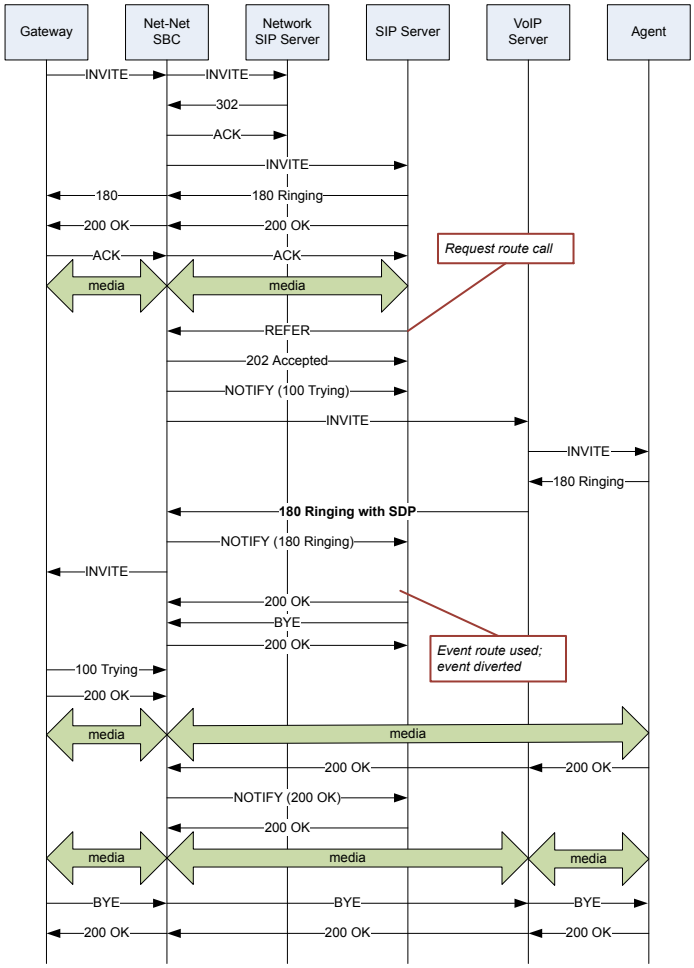
When you configure your Oracle USM to support REFER call transfers, you can enable it to send a NOTIFY message after it has sent either a 202 Accepted or sent a 180 Ringing message. If your network contains elements that comply with RFC 5589, and so expect the NOTIFY message after the 202 Accepted and each provisional 180 Ringing, you want to set the **refer-notify-provisional** to either **initial** or **all**, according to your needs.

Without this parameter changed from its default (**none**), the Oracle USM does not return send the NOTIFY until it receives the 200 OK response from the agent being called. If the time between the REFER and the NOTIFY exceeds time limits, this sequencing can cause the Oracle USM's NOTIFY to go undetected by devices compliant with RFC 5589. Failures during the routing process can result.

You can see how a sample call flow works without setting the **refer-notify-provisional** parameter.



When you compare the call flow above to the one depicting the scenario when the Oracle USM has the **refer-notify-provisional** changed from its default, you can see that the Oracle USM now response with a NOTIFY in response to the 202 Accepted and it sends another after the 180 Ringing.This causes the event to be diverted successfully.



Sample Messages

In compliance with RFC 5589, the NOTIFY message with 100 Trying as the message body looks like the sample below. Note that the expires value in the subscription state header is populated with a value that equals 2* TIMER C, where the default value of TIMER C is 180000 milliseconds.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
```

```
Call-ID: a84b4c76e66710
CSeq: 73 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: active;expires=360
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 100 Trying
```

Also in compliance with RFC 5589, the NOTIFY message with 180 Ringing as the message body looks like the sample below. Again, the expires value in the subscription state header is populated with a value that equals 2* TIMER C, where the default value of TIMER C is 180000 milliseconds.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
Call-ID: a84b4c76e66710
CSeq: 73 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: active;expires=360
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 180 Ringing
```

Also in compliance with RFC 5589, the NOTIFY message with 200 OK as the message body looks like the sample below.

```
NOTIFY sips:4889445d8kjtk3@atlanta.example.com;gr=723jd2d SIP/2.0
Via: SIP/2.0/TLS 192.0.2.4;branch=z9hG4bKnas432
Max-Forwards: 70
To: <sips:transferor@atlanta.example.com>;tag=1928301774
From: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>;tag=a6c85cf
Call-ID: a84b4c76e66710
CSeq: 74 NOTIFY
Contact: <sips:3ld812adkjw@biloxi.example.com;gr=3413kj2ha>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, tdialog
Event: refer
Subscription-State: terminated;reason=noresource
Content-Type: message/sipfrag
Content-Length: ...
SIP/2.0 200 OK
```

180 and 100 NOTIFY Configuration

You can apply the **refer-notify-provisional** setting to realms or to session agents. This section shows you how to apply the setting for a realm; the same steps and definitions apply to session agents.

If you do not want to insert NOTIFY messages into the exchanges that support REFER call transfers, you can leave the **refer-notify-provisional** set to **none**. This means that the Oracle

USM will send only the final result NOTIFY message. Otherwise, you want to choose one of the two settings described in the instructions below.

To enable 100 and 180 NOTIFY messages in REFER call transfers:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **refer-notify-provisional**—Choose from one of the following settings, where the Oracle USM:

   - **initial**—Sends an immediate 100 Trying NOTIFY, and the final result NOTIFY
   - **all**—Sends an immediate 100 Trying NOTIFY, plus a notify for each non-100 provisional messages the Oracle USM receives; and the final result NOTIFY

     ```
     ORACLE(realm-config)# refer-notify-provisional all
     ```

5. Save your work.

SIP REFER Re-Invite for Call Leg SDP Renegotiation

Enhancing the original implementation of SIP REFER termination introduced in Release S-C6.0.0, this change to Oracle USM behavior allows for SDP renegotiation between both parties of a transferred call.

Scenario

In a call transfer initiated by SIP REFER, a call is established between two user agents, UA-A and UA-B. UA-B then sends a REFER request to transfer the call to UA-C. The challenge is that UA-A and UA-B had already been communicating using mutually agreed-on codec, while UA-C might not be using an entirely different codec.

To solve this problem, the Oracle USM causes a new SIP session and new media session to be created between UA-A and UA-C. The Oracle USM removes any resources allocated for use between UA-A and UA-B, and then severs its connection with UA-B. The session between UA-A and UA-C continues.

Alterations to SIP REFER

The original implementation of the SIP REFER feature made available in Oracle USM Release S-C6.0.0 resulted in instances where SDP parameters were not being communicate properly. Issues arose when the Oracle USM maintained the original dialog with the user agent that did not support REFER and failed to communicate and SDP changes to that endpoint.

The alterations to SIP REFER made available in Oracle USM Release S-C(X)6.1.0M2 solve those issues. Now, the Oracle USM sends re-INVITE with the negotiated SDP to the user agent for which the Oracle USM performs the call transfer.

Implementation Details

This section describes the details of how the Oracle USM behaves in SIP REFER scenarios with the changes made in Oracle USM Release S-C(X)6.1.0M2. The Oracle USM makes the new call between Party A and Party C appear as though A were participating to allow the Oracle USM's natural media setup occur in the same way as if the REFER had actually been sent to A and A had sent a new INVITE.

When the Oracle USM receives a REFER request and determines it needs to handle it locally, it creates a new INVITE made to look like one from Party A. And the Oracle USM actually processes this INVITE as though it were from Party A. As a result, new SIP and new media sessions are created with new media ports for Parties A and C. When the INVITE to Party C receives a final response, the Oracle USM sends the result to Party B using a SIP NOTIFY request.

If the new INVITE succeeds, the old context and flows disappear and the new context and flows for the A-to-C connection remain in place. Because of the new media ports, the Oracle USM sends a re-INVITE to Party A, directing media to the new port and forwarded to Party C. Next, the original dialog with Party B needs to be terminated; if the Oracle USM has not received Party B's BYE, it will wait five second and then send Party B a BYE.

If the INVITE to Party C fails, the new SIP and media sessions are deleted as are the new context and flows. The Oracle USM treats Party A differently depending on whether or not a BYE was received from Party B. If a BYE was received from Party B. then the Oracle USM sends a BYE to Party A. If not, the original SIP and media sessions as well as the context and media flows remain in tact. This way, Party B can re-establish the call with Party A using a re-INVITE. In the case, the Oracle USM waits 32 second before sending a BYE.

If the Oracle USM receives a BYE while processing the INVITE to Party C, it sends a CANCEL message to Party C in an attempt to cancel the call. The BYE passes to Party B, and associated sessions, contexts, and flows terminate normally. Still, the Oracle USM waits for the final response to the INVITE to Party C. If the Oracle USM receives a successful response, it sends an ACK and then a BYE to terminate the abandoned call. If the Oracle USM receives an unsuccessful final response, it uses its normal response error handling processes. In either of these last two cases, all sessions, context, and flow are deleted.

Please note that the Oracle USM does not remove the a=sendonly attribute from the SDP it sends to Party A during the A-to-B call, and extra media ports are not allocated for the original media session.

SIP REFER with Replaces

To support enterprise and call center applications, the Oracle USM provides the ability for one party participating in a three-way call to request direct connectivity between the other two parties and to leave the call silently when that connectivity is established. SIP supports this function using the Replaces header in a REFER message, also known as REFER with Replaces.

The most common application of REFER with Replaces handling occurs in a high-level sequence like this:

1. The customer calls a customer service line and reaches—via the Oracle USM—an IVR/ACD (Interactive Voice Response system/Automatic Call Distribution system). In some architectures, these are two separate elements.
2. Based on the customer's selection from the menu of options, the IVR/ACD contacts an agent via the Oracle USM.

3. Since the ultimate goal is for the IVR/ACD to drop out of the path, it sends a REFER with Replaces to the Oracle USM. This message indicates the Oracle USM should replace the IVR/ACD endpoint in the call leg with the agent's endpoint.
4. The Oracle USM processes the REFER with Replaces, issuing ReINVITEs to the customer with the agent's parameters.
5. The IVR/ACD drops out of the media path once the bridged call between the customer and the agent is established.

   Note that direct media connectivity between endpoints must be possible in order for the REFER with Replaces to be carried out properly. For example, if both endpoints (such as the customer and agent from the example above) are behind the same firewall, direct media connectivity should be possible. However, if one endpoint is behind a firewall and the other is not, then direct media connectivity may not be possible.

   For licensing capacity purposes, note that a bridged session counts as a single call.

SIP REFER with Replaces Configuration

You enable SIP REFER with Replaces handling either in the realm configuration or in the session agent configuration. This section show you how to configure the feature for session agent, though the steps are the same for adding this feature to a realm.

To enable sending ReINVITES to a referred agent on an existing session/dialog:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name **refer-reinvite**. Then press Enter.

   ```
   ORACLE(session-agent)# options +refer-reinvite
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

SIP REFER-to-BYE

The Oracle USM's SIP REFER-to-BYE capability addresses situations when other network elements do not support the REFER method but do offer blind transfer in a SIP BYE request. The target number is encoded in a Reason header of the BYE request. In such cases, the Oracle USM terminates the REFER and passes the Refer-To number in a Reason header of the BYE.

You configure both SIP interfaces and SIP session agents with the refer-to-bye option to use this function:

- SIP interface—You add this ability to SIP interfaces facing the SIP elements that need to receive a BYE instead of a REFER. This setting only applies when the next hop is not a session agent.
- SIP session agent—The SIP session agent takes precedence over the SIP interface. You add this ability to SIP session agents that need to receive a BYE instead of a REFER. If the next hop SIP element—the remote target in the dialog—is a session agent, in other words, you need to configure the option for it. Note that when you use this option for SIP session agents, the SIP interface or realm on which the REFER is received takes precedence over the REFER-to-BYE capability.

When a REFER request arrives and the REFER-to-BYE capability applies, the Oracle USM responds to it with a 202 Accepted and sends a NOTIFY to terminate the implicit refer subscription. This NOTIFY contains a message/sipfrag body with SIP/2.0 200 OK. Upon receiving the response to this NOTIFY, the Oracle USM sends a BYE with an added Reason header (encoded with the Refer-To number) to the other end.

The network element that does not accept REFERs takes the BYE with the Reason header and issues a new initial INVITE that initiates transfer, which the Oracle USM sees as starting a new and independent session.

SIP Roaming

This section explains how to configure SIP roaming. SIP roaming lets subscribers move from one active SIP device to another (at the same site or multiple sites) and retain service at the last registering device.

Overview

The Oracle USM supports multiple active registrations for the same user. The softswitch makes decisions regarding the current location of the user and the handling of requests from devices that are not currently identified as the user location. When there are multiple NATs, the Oracle USM is still required to let the softswitch be able to differentiate it.

The Oracle USM's SIP roaming ability supports the following features:

- Multiple active registrations from the same user can be cached, allowing subscribers to move from one active SIP device to another (at the same site or multiple sites) and still retain service at the last registering device. With the SIP roaming feature, one person, using multiple devices, can be contacted at all of the devices. These multiple devices (with their unique contact information) register to indicate that they are available for anyone that wants to contact that one person.
- The Oracle USM can also inform network devices (such as softswitches) of private SIP device IPv4 addresses (endpoints) and the public firewall address of the user location.

Process Overview

Caller 1 wants to contact Person A. Caller 1 sends a message to persona@acmepacket.com, but Person A has configured more than one SIP-enabled device to accept messages sent to that address. These devices have unique addresses of desk@10.0.0.4 and phone2@10.0.0.5. Person A has desk@10.0.0.4 and phone2@10.0.0.5 registered with the Oracle USM for anything addressed to persona@acmepacket.com.

With the SIP roaming feature, the Oracle USM accepts and stores both registrations for persona@acmepacket.com. That way, when someone wants to get in touch with Person A, the

messages are sent to both devices (desk@10.0.0.4 and phone2@10.0.0.5) until Person A answers one of them. You do not need to configure your Oracle USM for this functionality; your Oracle USM automatically provides it.

Using Private IPv4 Addresses

In addition to supporting multiple registries, the Oracle USM (USM) can also distinguish user locations by their private IPv4 address and the IPv4 address of the public firewall. Using this information, the USM adds private endpoint and public firewall information to Contact headers.

For example, entering this information causes a Contact header that formerly appeared as the following:

```
Contact:<sip:0274116202@63.67.143.217>
```

to subsequently appear as the following:

```
Contact:<sip:0274116202@63.67.143.217;ep=192.168.1.10;fw=10.1.10.21>
```

The USM SIP proxy reads this information and populates the contact-endpoint and contact-firewall fields with the appropriate values.

Example 1 With a NAT Firewall

The Oracle USM (USM) SIP proxy is configured with the following changeable parameters:

- endpoint= IP address of the SIP UA
- useradd= IP address of the Firewall Public IP address or the source layer 3 IP address of Register message
- userport= IP address port number of the Firewall Public IP address or the source layer 3 IP address port of Register message
- USM address=63.67.143.217
- firewall public address=10.1.10.21
- firewall public address port=10000
- SIP endpoint behind firewall=192.168.1.10

SIP message Contact header:

```
Contact:<sip:0274116202@63.67.143.217; endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000; transport=udp>
```

Example 2 Without a NAT Firewall

The Oracle USM SIP proxy is configured with the following changeable parameters:

- useradd= IP address of the SIP UA or the source layer 3 IP address of Register message
- userport= IP address port number of the SIP UA or the source layer 3 IP address port of Register message
- Oracle USM address=63.67.143.217
- SIP endpoint=192.168.1.10
- SIP endpoint IP address port=5060

SIP message Contact header:

```
Contact:<sip:0274116202@63.67.143.217; useradd=192.168.1.10; userport=5060;
transport=udp>
```

For SIP, the softswitch responsibility is that the URI SD put in the Contact of the REGISTER message should be reflected in the 200-OK response to the REGISTER request. The Contact header of the response should have an expires header parameter indicating the lifetime of the registration.

The following example shows a Oracle USM Send:

```
Contact: <sep: 0274116202@63.67.143.217 endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000>;
```

The following examples shows the softswitch Respond:

```
Contact: <sep: 0274116202@63.67.143.217 endpoint=192.168.1.10;
useradd=10.1.10.21; userport=10000>;  expires=360
```

The contact field for endpoint and firewall parameters only appear in the following:

- Contact header of a REGISTER request sent from the Oracle USM to the softswitch server
- Contact header of a REGISTER response sent from the softwitch server to the Oracle USM
- Request-URI of an initial INVITE sent from the UT CSA server to the Oracle USM

An active endpoint is deleted when it does not register within the registration-interval setting or receives a 401 Unauthorized.

SIP Roaming Configuration

You can configure the SIP configuration's options parameter to indicate that you want to use the private IP address of the SIP device that the user is using and/or the public firewall address that identifies the location of the device. If defined, these options will be added as parameters to all Contact headers.

You can identify the endpoint and/or firewall information using the following options:

- contact-endpoint=<value> where <value> is the endpoint address or label
- contact-firewall=<value> where <value> is the firewall address or label

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure SIP config parameters. To view all SIP config parameters, enter a **?** at the system prompt.

4. Type **options** followed by a Space.

5. After the Space, type the information for an endpoint or a firewall, or both:

   ```
   contact-endpoint=″<label>
   contact-firewall=″<label>
   "contact-endpoint="<label>",contact-firewall="<label>""
   ```

6. Press Enter.

   For example, if you want your Oracle USM to add private endpoint and public firewall information to Contact headers, and you want to label this information as ep and fw, you would enter the following information in the ACLI.

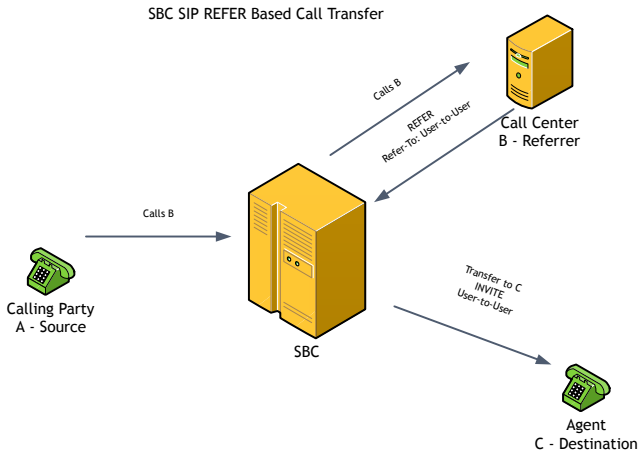
   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)# options "contact-endpoint="ep",contact-firewall="fw""
   ```

SIP REFER Call Transfer UUI Relay

The SIP REFER Call Transfer **User to User Information (UUI)** Relay option assists in the transfer of caller details through using the information in the "Refer-To" header in a new "User to User" header in the INVITE to the Referred-to party. This feature only works when the **refer-call-transfer** option is enabled on the realm or session agent where the REFER is received. This behavior change is enabled by default. This option can be used by a Call Center application to transfer a call with user information to an agent.

A new INVITE is sent with the information about the calling user to the agent by way of the "Refer-To" header using the **User to User Information.** The product variable needs to relay this UUI as a separate header in the INVITE message while transferring the call to the destination. In this scenario, the product terminates the REFER message having captured the UUI and then sends an INVITE with the UUI to the agent. This feature works only when the **refer-call-transfer** option is enabled on the Realm or Session Agent where REFER is received. By default this option is enabled and thus needs to be specifically disabled in the configuration if not wanted.

The following illustration shows the path of the call from the source to the destination by way of the product utilizing the **User to User Information** relay



SIP REFER UUI Relay

Refer-To Header

The Oracle USM supports the format of **User-to-User** in the "Refer-To" header in the REFER message. The following example shows the new text after the IP address and port in the second to last line.

```
REFER sip:7325550000@192.168.28.10:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.29.1:5060;branch=z9hG4bK-10659-1-4
From: 7325550000 <sip:7325550000@192.168.29.1:5060>;tag=1
To: 7321110000 <sip:732111000@192.168.28.19:5060>;tag=1
Call-ID: 1-10679@192.168.28.1
...
Refer-To: <sip:9785550000@192.168.29.1:6062?User-to-
User=56a390f3d2b7310023a%3Bencoding%3Dhex%3Bpurpose%3Disdn-
interwork%3Bcontent%3Disdn-uui
Content-Length:0
```

User to User Parameter in INVITE

If the UUI is received in the "Refer-To" header in the REFER message, the Oracle USM adds the new UUI header in the INVITE message to the destination party. The system encodes the INVITE message as shown in the following example. All the escape characters received in the UUI parameter converts to plain text and all the UUI header parameters are relayed. In the example below, note the new UUI content below **Content-Length** and in the **Supported** fields.

```
INVITE sip:9785550000@192.168.29.1:6062 SIP/2.0
Via: SIP/2.0/UDP 192.168.29.20:5060;branch=z9hG4bKgkm8110090jmlvcfk80.1
From: 7321110000 <sip:7321110000@192.168.28.1:5060>;tag=1
To: <sip:9785550000@192.168.29.1:6062>
Call-ID: 537e3302f02bbb0ddd8b8d7538f8b33030@192.168.28.1
```

```
CSeq: 2 INVITE
Contact:<sip:sipp@192.168.29.20:5060;transport=UDP>
Session-Expires: 3600;refresher=uac
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 135
User-to-User:56a390f3d2b7310023a;encoding=hex;purpose=isdn-
interwork;content=isdn-uui
Supported: timer,uui
Referred-By:<sip.7325550000@192.168.29.1:5060>
Route:<sip:9785550000@core2:6020,lr>

v=0
o=user153655765 2353687637 IN IP$ 192.168.28.1
s=-
c=IN IP4 192.168.28.1
t= 0 0
m=audio 8000 RTP/AVP 8
a=rtpmap 8 PCMA/8000
```

User to User Header in REFER

The product supports the REFER header that replaces the default header with the UUI header in an attended call transfer.

```
Aug 4 10:09:10.233 On [256:0]192.168.12.1:5060 received from 192.168.12.2:5060
REFER sip:1000@192.168.12.1:5060 SIP/2.0
Via: SIP/2.0/udp 192.168.12.2:5060;branch=z9hG4bK-bob2alice-9
Max-Forwards: 10
From: bob <sip:2000@acme.com>;tag=bob-0
To: alice <sip:1000@acme.com>;tag=alice-1
Call-ID: 1-192.168.11.2
CSeq: 3 REFER
Refer-To: <sip:3000@acme.com?Replaces=1-192.168.12.2%3Bto-tag
%3Dcarol2bob-0%3Bfrom-tag%3Dbob2carol-2&
User-to-User=56a390f3d2b7310023a%3Bencoding%3Dhex%3Bpurpose%3Disdn-interwork
%3Bcontent%3Disdn-uui>
```

INVITE Message Details

The Oracle USM adds the UUI parameter to the **Supported** header in the INVITE message to the **Referred-to** party. The Oracle USM encodes only one instance of the UUI header in the INVITE message to the **Referred-to** party. The Oracle USM will only process the first UUI parameter received in the "Refer-To" header. The Oracle USM supports a maximum of 128 octets hex content in the UUI header field, excluding parameters. The overall length of the header complies with RFC 3261. If the header length exceeds the maximum, the Oracle USM discards the UUI header and cannot relay the call.

SIP REFER Call Transfer UUI Relay Configuration

The SIP REFER Call Transfer **User to User Information** Relay is enabled by default. To disable UUI relaying on a configured realm:

1. Access the **realm-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **options**—Set the options parameter by typing options, a space, **disable-refer-to-uui=yes**. You may prepend the option name with a plus to add and not replace this option to the existing **realm-config** option list.

```
ORACLE#options +disable-refer-to-uui=yes
```

4. Type **done** to save your configuration.

Embedded Header Support

This section explains how to configure embedded header support. The Oracle USM supports methods of extracting an embedded P-Asserted-Identity header from a contact header to support E911 when integrated with certain vendor's systems. See RFC 3455 *Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)* for more information.

The embedded header support feature watches for a specified embedded header contained in a Contact header received in a 3XX message. When the specified embedded header is found, the full <header=value> pair is inserted as a unique header in a redirected INVITE message that exits the Oracle USM. If the outgoing INVITE message were to contain the specified header, regardless of the use of this feature, the value extracted from the 3XX message replaces the INVITE message's specified header value.

If an incoming Contact header in a 3XX message looks like:

```
Contact: <ESRN@IPv4_Intrado_GW;user=phone?P-Asserted-Identity=%3Csip:+1-
ESQK@IPv4_My_EAG;user=phone%3E>
```

Then, if you configure your Oracle USM to parse for the embedded P-Asserted-Identity header to write as a unique header in the outgoing invite message, the outgoing INVITE and P-Asserted-Identity headers will look like:

```
INVITE SIP: ESRN@IPv4_Intrado_GW;user=phone
P-Asserted-Identity: +1-ESQK@IPv4_My_EAG;user=phone
```

Embedded Header Support Configuration

Embedded header support is enabled in the session agent configuration.

To configure embedded header support:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. Select the session agent where you want this feature.

```
ORACLE(session-agent)# select
<hostname>:
1: asd          realm=      ip=1.0.0.0
2: SIPSA        realm=      ip=10.10.102.1
selection:2
ORACLE(session-agent)#
```

5. **request-uri-headers—Enter a list of embedded headers extracted from the Contact header that will be inserted in the re INVITE message.** To configure this parameter for multiple headers, enclose the headers in double quotes and separate them with spaces. This completes the configuration of embedded header support.

```
ORACLE(session-agent)# request-uri-headers P-Asserted-Identity
```

Static SIP Header and Parameter Manipulation

This section explains the SIP header and parameter manipulation feature, which lets the Oracle USM add, modify, and delete SIP headers and parts of SIP headers called SIP header elements. SIP header elements are the different subparts of the header, such as the header value, header parameter, URI parameter and so on (excluding the header name).

To enable the SIP header and parameter manipulation functionality, you create header manipulation rulesets in which you specify header manipulation rules, as well as optional header element rules that operate on specified header elements. You then apply the header manipulation ruleset as inbound or outbound for a session agent or SIP interface.

Header Manipulation Rules

Header manipulation rules operate on the header you specify when you configure the rule. A header manipulation rule can also be configured with a list of element rules, each of which would specify the actions you want performed for a given element of this header.

Header Element Rules

Header element rules perform operations on the elements of a header. Header elements include all subparts of a header; excluding the header name. For example, header value, header parameter, URI parameter, and so on.

About SIP Header and Parameter Manipulation

Using the SIP header manipulation ruleset, you can cause the Oracle USM to:

- Delete a header based on header name match.
- Delete a header based on header name match as well as header value match.
- Add a header.
- Modify the elements of a header (by configuring header element rules):
  Add an element to a header.

  For example, add a parameter to a header or add a URI parameter to the URI in a header.

  Delete an element from a header.

For example, delete a parameter from a header or delete a URI parameter from the URI in a header.

Modify an element of a header.

For example, replace a FQDN with an IPv4 address in a header or replace the value of a parameter in the header.

Delete a message body part

For example, delete the body part if the Content-Type is application/ISUP.

HMR $LOCAL_PORT for Port Mapping

When you configure SIP HMR and set an element-rule's new-value parameter to $LOCAL_PORT, the Oracle USM maps this value to the real port it uses for each signaling exchange.

Role in Trunk Group URI Feature

SIP header and parameter manipulation plays a role in the trunk group URI feature. You need to set the new-value parameter to one of the trunk group values when configuring SIP header rules, if using this feature. (In addition you can configure session agents and session agents groups on the Oracle USM to insert trunk group URI parameters in the SIP contact header.

For all trunk group URI support, you must set the appropriate parameters in the SIP header manipulation configuration and in the session agent or session agent group configurations.

For trunk group URI support, the SIP header and parameter manipulation configuration tells the Oracle USM where and how to manipulate the SIP message to use originating (access) and terminating (egress) trunk group URI parameters.

SIP Header and Parameter Manipulation Configuration

This section explains how to configure SIP header and parameter manipulation. First you create a SIP header manipulation ruleset, then the header manipulation rules and optional header element rules you want that ruleset to contain. You then configure a session agent or a SIP interface to use the SIP header and parameter manipulation ruleset in the inbound and outbound directions.

Creating SIP Header Manipulation Rulesets

To configure the SIP header manipulation ruleset:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-manipulation** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. **name**—Enter the name you want to use for this ruleset.

5. **header-rules**—Define the header manipulation rules you want to include in this ruleset.

   Type header-rules and press Enter.

   ```
   ORACLE(sip-manipulation)# header-rules
   ORACLE(sip-header-rules)#
   ```

   name—Enter the name of the header to which this rule applies. (The name you enter here must match a header name.)

   This is a case-insensitive string that is compared to the header name for matching. You need to create a rule using the long form of the header name and a rule using the compact form of the header name.

   > **Note:**
   >
   > The Request-URI header is identified as request-uri.

   action—Enter the action you want applied to the header specified in the name parameter. The default value is none. Valid options are:

   - **add**—Add a new header, if that header does not already exist.
   - **delete**—Delete the header, if it exists.
   - **manipulate**—Elements of this header will be manipulated according to the element rules configured.
   - **store**—Store the header.
   - **none**—No action to be taken.

     match-value—Enter the value to be matched (only an exact match is supported) with a header value. The action you specify is only performed if the header value matches.

     msg-type—Enter the message type to which this header rule applies. The default value is any. Valid options are:

   - **any**—Both Requests and Reply messages
   - **request**—Request messages only
   - **reply**—Reply messages only

     Type show to display the header rule configuration values.

6. **element-rules**—Define the element rules you want to use to be performed on the elements of the header specified by the header rule.

   Type element-rules and press Enter.

   ```
   ORACLE(sip-header-rules)# element-rules
   ORACLE(sip-element-rules)#
   ```

   name—Enter the name of the element to which this rule applies.

> **Note:**
>
> The name parameter usage depends on the element type you enter in step 6. For uri-param, uri-user-param, and header-param it is the parameter name to be added, replaced, or deleted. For all other types, it serves to identify the element rule and any name can be used.

type—Enter the type of element on which to perform the action. The default value is none. Valid options are:

- **header-value**—Enter value of the header.
- **header-param-name**—Header parameter name.
- **header-param**—Parameter portion of the header.
- **uri-display**—Display of the SIP URI.
- **uri-user**—User portion of the SIP URI.
- **uri-host**—Host portion of the SIP URI.
- **uri-port**—Port number portion of the SIP URI.
- **uri-param-name**—Name of the SIP URI param.
- **uri-param**—Parameter included in the SIP URI.
- **uri-header-name**—SIP URI header name
- **uri-header**—Header included in a request constructed from the URI.
- **uri-user-param**—User parameter of the SIP URI.

action—Enter the action you want applied to the element specified in the name parameter, if there is a match value. The default value is none. Valid options are:

- **none**—No action is taken.
- **add**—Add a new element, if it does not already exist.
- **replace**—Replace the elements.
- **delete-element**—Delete the specified element if it exists. Based on the match value if entered in step 6f.
- **delete-header**—Delete the specified header, if it exists.
- **store**—Store the elements.

match-val-type—Enter the type of value that needs to be matched to the match-field entry for the action to be performed. The default value is ANY. Valid options are:

- **IP**—Element value in the SIP message must be a valid IP address to be compared to the match-value field entry. If the match-value field is empty, any valid IP address is considered a match. If the element value is not a valid IP address, it is not considered a match.
- **FQDN**—Element value in the SIP message must be a valid FQDN to be compared to the match-value field entry. If the match-value field is empty, any valid FQDN is considered a match. If the element value is not a valid FQDN, it is not considered a match.
- **ANY**—Element value in the SIP message is compared with the match-value field entry. If the match-value field is empty, all values are considered a match.

match-value-—Enter the value you want to match against the element value for an action to be performed.

new-value-—Enter the value for a new element or to replace a value for an existing element. You can enter an expression that includes a combination of absolute values, pre-defined parameters, and operators.

- Absolute values, with which you can use double quotes for clarity. You must escape all double quotes and back slashes that are part of an absolute value, and enclose the absolute value in double quotes.

  For example:

  sip:”+$TRUNK_GROUP+”.$TRUNK_GROUP_CONTEXT

- Pre-defined parameters always start with a $. Valid pre-defined parameters are:

Parameter	Description
$ORIGINAL	Original value of the element is used.
$LOCAL_IP	IP address of the SIP interface on which the message was received for inbound manipulation; or sent on for outbound manipulation.
$REMOTE_IP	IP address the message was received from for inbound manipulation; or being sent to for outbound manipulation.
$REMOTE_VIA_HOST	Host from the top Via header of the message is used.
$TRUNK_GROUP	Trunk group is used.
$TRUNK_GROUP_CONTEXT	Trunk group context is used.

- Operators are:

Operator	Description
+	Append the value to the end. For example: acme”+”packet generates acmepacket
+^	Prepends the value. For example: acme”+^”packet generates packetacme
-	Subtract at the end. For example: 112311”-”11 generates 1123
-^	Subtract at the beginning. For example: 112311”-^”11 generates 2311

Examples of entries for the **new-value** field.

```
$ORIGINAL+acme
$ORIGINAL+"my name is john"
$ORIGINAL+"my name is \"john\""
$ORIGINAL-^781+^617
```

Type show to display the element rule configuration values.

Type done to save them.

Repeat steps 6b through 6j to create additional rules.

Type exit to return to the header-rules parameters.

7. **methods-—**Enter the SIP method names to which you want to apply this header rule. If entering multiple method names, separate them with commas. For example:

   ```
   INVITE,ACK,BYE
   ```

   This field is empty by default. If you leave the method field empty, the header-rule is applied to all methods.

8. Type **exit** to return to the sip-manipulation level.
9. Save your work using the ACLI **done** command.
10. If you want to save this configuration, exit out of configuration mode and type **save-config**.

Configuring a Session Agent

You can configure a session agent to use the SIP header manipulation ruleset.

To configure a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **in-manipulationid**—Enter the name of the SIP header manipulation ruleset you want to apply to inbound SIP packets.

   ```
   ORACLE(session-agent)# in-manipulationid route-stripper
   ```

5. **out-manipulationid**—Enter the name of the SIP header manipulation ruleset you want to apply to outbound SIP packets.

   ```
   ORACLE(session-agent)# out-manipulationid route-stripper
   ```

6. Save your work using the ACLI **done** command.
7. If you want to save this configuration, exit out of configuration mode and type **save-config**.

Configuring a SIP Interface

You can configure a interface to use a SIP header manipulation ruleset.

To configure a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **in-manipulationid**—Enter the name of the SIP header manipulation ruleset you want to apply to SIP packets in the ingress direction.

   ```
   ORACLE(sip-interface)# in-manipulationid
   ```

5. **out-manipulationid**—Enter the name of the SIP header manipulation ruleset you want to apply to SIP packets in the egress direction.

   ```
   ORACLE(sip-interface)# out-manipulationid
   ```

6. Save your work using the ACLI **done** command.
7. If you want to save this configuration, exit out of configuration mode and type **save-config**.

Example 1 Stripping All Route Headers

This example explains how to strip all route headers from a SIP packet. First, you create a header manipulation ruleset, in the example it is called route-stripper. Then you configure the list of header manipulation rules you need to strip route headers. In this case, you only need one rule named Route (to match the Route header name) with the action set to Delete.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)# name route-stripper
ORACLE(sip-manipulation)# header-rules
ORACLE(sip-header-rules)# name Route
ORACLE(sip-header-rules)# action Delete
ORACLE(sip-header-rules)# done
header-rule
        name                            Route
        action                          delete
        match-value
        msg-type                        any
ORACLE(sip-header-rules)# ex
ORACLE(sip-manipulation)# done
sip-manipulation
        name                            route-stripper
        header-rule
                name                            Route
                action                          delete
                match-value
                msg-type                        any
```

Example 2 Stripping an Existing Parameter and Adding a New One

This example explains how to strip the user parameter from the Contact header URI and add the acme parameter with value as LOCAL IP, only for requests. First you create a header manipulation ruleset, in the example it is called param-stripper1. You then configure a list of header rules you need. In this case, you only need one rule named Contact (to match the Contact header name), with action set to manipulate (indicating the elements of this header would be manipulated). Next, you configure a list of element rules for the Contact header rule.

In this case you configure two element rules; one to strip the uri parameter user (the rule name user matches the param name user) and the other to add the uri parameter acme (the rule name acme matches the param name acme).

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)# name param-stripper1
ORACLE(sip-manipulation)# header-rules
ORACLE(sip-header-rules)# name Contact
ORACLE(sip-header-rules)# action manipulate
ORACLE(sip-header-rules)# msg-type request
ORACLE(sip-header-rules)# element-rules
ORACLE(sip-element-rules)# name user
ORACLE(sip-element-rules)# type uri-param
ORACLE(sip-element-rules)# action delete-element
ORACLE(sip-element-rules)# done
element-rule
        name                            user
        type                            uri-param
        action                          delete-element
        match-val-type                  any
        match-value
        new-value
ORACLE(sip-element-rules)# name acme
ORACLE(sip-element-rules)# action add
ORACLE(sip-element-rules)# type uri-param
ORACLE(sip-element-rules)# new-value "$LOCAL_IP"
ORACLE(sip-element-rules)# done
element-rule
        name                            acme
        type                            uri-param
        action                          add
        match-val-type                  any
        match-value
        new-value                       "$LOCAL_IP"
ORACLE(sip-element-rules)# ex
ORACLE(sip-header-rules)# done
header-rule
        name                            Contact
        action                          manipulate
        match-value
        msg-type                        request
        element-rule
                name                            user
                type                            uri-param
                action                          delete-element
                match-val-type                  any
                match-value
                new-value
        element-rule
                name                            acme
                type                            uri-param
                action                          add
                match-val-type                  any
                match-value
                new-value                       "$LOCAL_IP"
ORACLE(sip-header-rules)# ex
ORACLE(sip-manipulation)# done
sip-manipulation
        name                            param-stripper1
```

```
        header-rule
                name                            Contact
                action                          manipulate
                match-value
                msg-type                        request
                element-rule
                        name                            user
                        type                            uri-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
element-rule
                name                            acme
                type                            uri-param
                action                          add
                match-val-type                  any
                match-value
                new-value                       "$LOCAL_IP"
```

For example, if the IP address of the SIP interface ($LOCAL_IP) is 10.1.2.3 and the Oracle USM receives the following Contact header:

```
Contact: <sip:1234@10.4.5.6;user=phone>
```

The header rule is applied to strip the user parameter from the Contact header URI and add the acme parameter with the value 10.1.2.3:

```
Contact: <sip:1234@10.4.5.6;acme=10.1.2.3>
```

SIP HMR (Header Manipulation Rules)

SIP header manipulation can also be configured in a way that makes it possible to manipulate the headers in SIP messages both statically and dynamically. Using this feature, you can edit response headers or the Request-URI in a request, and change the status code or reason phrase in SIP responses.

Static SIP Header and Parameter Manipulation allows you to set up rules in your Oracle USM configuration that remove and/or replace designated portions of specified SIP headers. SIP HMR allows you to set up dynamic header manipulation rules, meaning that the Oracle USM has complete control over alterations to the header value. More specifically:

- The Oracle USM can search header for dynamic content or patterns with the header value. It can search, for example, for all User parts of a URI that begin with 617 and end with 5555 (e.g., 617...5555).
- The Oracle USM can manipulate any part of a patterns match with any part of a SIP header. For example, 617 123 5555 can become 617 231 5555 or 508 123 0000, or any combination of those.

To provide dynamic header manipulation, the Oracle USM uses regular expressions to provide a high degree of flexibility for this feature. This allows you to search a specific URI when you do not know that value of the parameter, but want to use the matched parameter value as the header value. It also allows you to preserve matched sections of a pattern, and change what you want to change.

You can apply header manipulation to session agents, SIP interfaces, and realms. You do so by first setting up header manipulations rules, and then applying them in the configurations where

they are needed. Within the header manipulation rules, there are sets of element rules that designate the actions that need to be performed on a given header.

Each header rule and each element rule (HMR) have a set of parameters that you configure to identify the header parts to be manipulated, and in what way the Oracle USM is to manipulate them. These parameters are explained in detail, but the parameter that can take regular expression values is **match-value**. This is where you set groupings that you want to store, match against, and manipulate.

Generally, you set a header rule that will store what you want to match, and then you create subsequent rules that operate on this stored value. Because header rules and element rules are applied sequentially, it is key to note that a given rule performs its operations on the results of all the rules that you have entered before it. For example, if you want to delete a portion of a SIP header, you would create Rule 1 that stores the value for the purpose of matching, and then create Rule 2 that would delete the portion of the header you want removed. This prevents removing data that might be used in the other header rules.

Given that you are using regular expression in this type of configuration, this tightly sequential application of rules means that you must be aware of the results to be yielded from the application of the regular expressions you enter. When you set a regular expression match value for the first rule that you enter, the Oracle USM takes the results of that match, and then a second rule might exist that tells the Oracle USM to use a new value if it the second rule's match value finds a hit (and only 10 matches, 0-9, are permitted) for the results (yield) from applying the first rule.

Consider the example of the following regular expression entry made for a **match-value** parameter:'Trunk(.+)', which might be set as that match value in the first rule you configure. Given a SIP element rule called uri-param and the param-name tgid, it can yield two values:

- Grouping 0—The entire matching string (Trunk1)
- Grouping 1—The grouping (1)

In turn, these groupings can be referenced in an element rule by using this syntax:

```
$<header rule name >.$<element rule name.$<value>
```

Additional syntax options that can be used with this feature are:

- $headerName['['index']']
- $headerName['['index']'][.$index]
- $headerName['['index']'][.$elementName]
- $headerName['['index']'][.$elementName][.$index]

Guidelines for Header and Element Rules

Header rules and element rules share these guidelines:

- References to groupings that do not exist result in an empty string.
- References to element rule names alone result in a Boolean condition of whether the expression matched or not.
- A maximum of ten matches are allowed for a regular expression. Match 0 (grouping 0) is always the match of the entire matching string; subsequent numbers are the results for other groups that match.

Precedence

The Oracle USM applies SIP header rules in the order you have entered them. This guards against the Oracle USM removing data that might be used in the other header rules.

This ordering also provides you with ways to use manipulations strategically. For example, you might want to use two rules if you want to store the values of a regular expression. The first rule would store the value of a matched regular expression, and the second could delete the matched value.

In addition to taking note of the order in which header rules are configured, you now must also configure a given header rule prior to referencing it. For example, you must create Rule1 with the action store for the Contact header BEFORE you can create Rule2 which uses the stored value from the Contact header.

Duplicate Header Names

If more than one header exists for the header name you have configured, the Oracle USM stores the value where it can be referenced with the optional syntax $<header rule name>[index]. Additional stored header values are indexed in the order in which they appear within the SIP message, and there is no limit to the index.

Possible index values are:

- ~ — The Oracle USM references the first matching header
- * — The Oracle USM references all headers
- ^ — The Oracle USM references the last stored header in the header rule

Performing HMR on a Specific Header

HMR has been enhanced so that you can now operate on a specific instance of a given header. The syntax you use to accomplish this is similar to that you used to refer to a specific header rule stored value instance.

Using the header-name parameter, you can now add a trailing [<index>] value after the header name. This [<index>] is a numerical value representing the specific instance of the header on which to operate. However, the Oracle USM takes no action if the header does not exist. You can also use the caret (^) to reference the last header of that type (if there are multiple instances)

The count for referencing is zero-based, meaning that the first instance of the header counts as 0.

Note that the header instance functionality has no impact on HMR's add action, and you cannot use this feature to insert headers into a specific location. Headers are added to the end of the list, except that Via headers are added to the top.

Multiple SIP HMR Sets

In general you use SIP HMR by configuring rules and then applying those rules to session agents, realms, or SIP interfaces in the inbound or outbound direction. In addition, the Oracle USM has a set method for how certain manipulation rules take precedence over others. For instance, inbound SIP manipulation rules defined in a session agent take precedence over any

configured for a realm, and the rules for a realm take precedence over SIP interface manipulation rules.

The multiple SIP HMR feature gives you the ability to:

- Apply multiple inbound and outbound manipulations rules to a SIP message
- Provision the order in which the Oracle USM applies manipulation rules

The **action** parameter in the header rules configuration now takes the value sip-manip. When you set the parameter to sip-manip, you then configure the **new-value** parameter with the name of a SIP manipulation rule that you want to invoke. The values for the **match-value**, **comparison-type**, and **methods** parameters for invoked rule are all supported. This means that the manipulation defined by the rules identified in the **new-value** parameter are carried out when the values for the **match-value**, **comparison-type**, and **methods** parameters are true.

The relationship between manipulation rules and manipulation rule sets is created once you load your configuration, meaning that the order in which you enter them does not matter. It also means that the Oracle USM cannot dynamically perform validation as you enter rules, so you should use the ACLI **verify-config** command to confirm your manipulation rules contain neither invalid nor circular references. Invalid references are those that point to SIP manipulation rules that do not exist, and circular references are those that create endless loops of manipulation rules being carried out over and over. If you load a configuration exhibiting either of these issues, the Oracle USM forces the action value for the rule to **none** and the rule will not be used.

MIME Support

Using the SIP HMR feature set, you can manipulate MIME types in SIP message bodies. While you can manipulate the body of SIP messages or a specific content type using other iterations of SIP HMR, this version gives you the power to change the MIME attachment of a specific type within the body by using regular expressions. To achieve this, you use the **find-replace-all** action type, which enables the search for a particular string and the replacement of all matches for that type. Although you use **find-replace-all** to manipulate MIME attachments, it can also be used to achieve other goals in SIP HMR.

Note that using **find-replace-all** might consume more system resources than other HMR types. Therefore this powerful action type should only be used when another type cannot perform the type of manipulation you require.

Find and Replace All

To manipulate a particular portion of the MIME attachment, for example when removing a certain attribute within the content type of application/sdp, the Oracle USM (USM) would need to search the content multiple times because:

- SDP can have more than one media line
- The SIP message body can contain more than one application/sdp.

The **find-replace-all** action type works for SIP header rules and for element rules. You can use it for all manipulation types from the entire header value, to the URI specific parameters, to MIME attachment.

For this action type, it does not matter what you configure the comparison type, which is atypical for actions types, as the comparison type is vital to the others. **Find-replace-all**, however, binds the comparison type to the pattern rule. Thus, the USM treats the match value as a regular expression, and it ignores any configured comparison type value in favor of the

pattern rule. This type of action is both a comparison and action: For each regular expression match within the supplied string, the USM substitutes the new value for that match. Yet if you want to replace a certain portion of the regular expression and not the entire matched expression, you need to use a subgroup of expressions and the right syntax to indicate the sub-group replacement index.

You can indicate the sub-group replacement syntax by adding the string [[:n:]] to the end of the regular expression—where n is a number between 0 and 9. For example, given the following settings:

- action=find-replace-all
- match-value=sip:(user)@host[[:1:]]
- new-value=bob

you create a new rule to replace only the user portion of the URI that searches for the regular expression and replaces all instances of the user subgroup with the value bob.

Taking advantage of the **find-replace-all**'s recursive nature, you can replace all the 0 digits in a telephone number with 1:

- action=find-replace-all
- match-value=0
- new-value=1

So for the user portion of a URI—or for any other string—with a value 1-781-308-4400 would be replaced as 1-781-318-4411.

If you leave the **new-value** parameter blank for **find-replace-all**, the USM replaces the matched sub-group with an empty string—an equivalent of deleting the sub-group match. You can also replace empty sub-groups, which is like inserting a value within the second sub-group match. For example, user()@host.com[[:1:]]with a configured **new-value** _bob yields user_bob@host.com.

When you use **find-replace-all**, you cannot use the following **parameter-type** values: **uri-param-name**, **uri-header-name**, and **header-param-name**. These values are unusable because the USM only uses case-sensitive matches for the match-value to find the parameter name within the URI. Since it can only be found by exact match, the USM does not support finding and replacing that parameter.

Escaped Characters

SIP HMR's support for escaped characters allows for searches for values you would be unable to enter yourself. Because they are necessary to MIME manipulation, support for escaped characters now includes:

- \f
- \n
- \r
- \t
- \v

New Reserved Word

To allow you to search for carriage returns an new lines, the SIP HMR MIME feature also adds support for the reserved word $CRLF. Because you can search for these value and replace them, you also must be able to add them back in when necessary. Configuring $CRLF in the **new-value** parameter always resolves to /r/n, which you normally cannot otherwise enter through the ACLI.

About the MIME Value Type

Introduced to modify the MIME attachment, SIP HMR supports a **mime** value for the **type** parameter in the element rules configuration. Like the **status-code** and **reason-phrase** values, you can only use the **mime** type value against a specific header—which in this case, is Content-Type.

When you set the element rule type to **mime**, you must also configure the **parameter-name** with a value. This step is a requirement because it sets the content-type the Oracle USM manipulates in a specific part of the MIME attachment. You cannot leave this parameter blank; the Oracle USM does not let you save the configuration if you do. When you use the **store** action on a multi-part MIME attachment that has different attachment types, the Oracle USM stores the final instance of the content-type because it does not support storing multiple instances of element rule stored values.

In the event you do not know the specific content-type where the Oracle USM will find the **match-value**, you can wildcard the **parameter-name** by setting with the asterisk (*) as a value. You cannot, however, set partial content-types (i.e., application/*). So configured, the Oracle USM loops through the MIME attachment's content types.

You can set the additional **action** types listed in this table with the described result:

Action Type	Description
delete-element	Removes the matched mime-type from the body. If this is the last mime-type within in message body, the Oracle USM removes the Content-Type header.
delete-header	Removes all body content and removes the Content-Type header.
replace	Performs a complete replacement of the matched mime-type with the new-value you configure.
find-replace-all	Searches the specifies mime-type's contents and replaces all matching regular expressions with the new-value you configure
store	Stores the final instance of the content-type (if there are multi-part MIME attachments of various attachment types)
add	Not supported

MIME manipulation does not support manipulating headers in the individual MIME attachments. For example, the Oracle USM cannot modify the Content-Type given a portion of a message body like this one:

```
--boundary-1
Content-Type: application/sdp
v=0
o=use1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=audio 10000 RTP/AVP 8
```

```
a=rtpmap:8 PCMA/8000/1
a=sendrecv
a=ptime:20
a=maxptime:200
```

Back Reference Syntax

You can use back reference syntax in the **new-value** parameter for header and element rules configurations. Denoted by the use of $1, $2, $3, etc. (where the number refers to the regular expression's stored value), you can reference the header and header rule's stored value without having to use the header rule's name. It instead refers to the stored value of this rule.

For example, when these settings are in place:

- header-rule=changeHeader
- action=manipulate
- match-value=(.+)([^;])

you can set the **new-value** as sip:$2 instead of **sip:$changeHeader.$2**.

You can use the back reference syntax for:

- Header rule **action**s **manipulate** and **find-replace-all**
- Element rule **action**s **replace** and **find-replace-all**

Using back reference syntax simplifies your configuration steps because you do not need to create a store rule and then manipulate rule; the manipulate rule itself performs the store action if the **comparison-type** is set to **pattern-rule**.

Notes on the Regular Expression Library

In the regular expression library, the dot (.) character no longer matches new lines or carriage returns. Conversely, the not-dot does match new lines and carriage returns. This change provides a safety mechanism preventing egregious backtracking of the entire SIP message body when there are no matches. Thus, the Oracle USM reduces backtracking to a single line within the body. In addition, there is now support for:

Syntax	Description
\s	Whitespace
\S	Non-whitespace
\d	Digits
\D	Non-digits
\R	Any \r, \n, \r\n
\w	Word
\W	Non-word
\A	Beginning of buffer
\Z	End of buffer
\	Any character including newline, in the event that the dot (.) is not

In addition, there is:

- Escaped character shortcuts (\w\W\S\s\d\D\R) operating inside brackets [...]

SIP Message-Body Separator Normalization

The Oracle USM supports SIP with Multipurpose Internet Mail Extension (MIME) attachments — up to a maximum payload size of 64KB — and has the ability to allow more than the required two CRLFs between the SIP message headers and the multipart body's first boundary. The first two CRLFs that appear in all SIP messages signify the end of the SIP header and the separation of the header and body of the message, respectively. Sometimes additional extraneous CRLFs can appear within the preamble before any text.

The Oracle USM works by forwarding received SIP messages regardless of whether they contain two or more CRLFs. Although three or more CRLFs are legal, some SIP devices do not accept more than two.

The solution to ensuring all SIP devices accept messages sent from the Oracle USM is to strip all CRLFs located at the beginning of the preamble before the appearance of any text, ensuring that there are no more than two CRLFs between the end of the last header and the beginning of the body within a SIP message. You enable this feature by adding the new stripPreambleCrlf option to the global SIP configuration.

To enable the stripping of CRLFs in the preamble:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE#(configure)
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE#(session-router)
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name stripPreambleCrlf with a plus sign.

   ```
   ORACLE(sip-config)# options +stripPreambleCrlf
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the global SIP configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

SIP Header Pre-Processing HMR

By default, the Oracle USM (USM) performs in-bound SIP manipulations after it carries out header validation. Adding the **inmanip-before-validate** option in the global SIP configuration allows the USM to perform HMR on received requests prior to header validation. Because there are occaisional issues with other SIP implementations—causing invalid headers to be used in messages they send to the USM—it can be beneficial to use HMR to remove or repair these faulty headers before the request bearing them are rejected.

When configured to do so, the USM performs pre-validation header manipulation immediately after it executes the top via check. Inbound SIP manipulations are performed in order of increasing precedence: SIP interface, realm, and session agent.

The fact that the top via check happens right before the USM carries out pre-validation header manipulations means that you cannot use this capability to repairs the first via header if it is indeed invalid. If pre-validation header manipulation were to take place at another time during processing, it would not be possible to use it for SIP session agents. The system learns of matching session agents after top via checking completes.

For logistical reasons, this capability does not extend to SIP responses. Inbound manipulation for responses cannot be performed any sooner that it does by default, a time already preceding any header validation.

To enable SIP header pre-processing:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE#(configure)
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE#(session-router)
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name inmanip-before-validate with a plus sign.

   ```
   ORACLE(sip-config)# options +inmanip-before-validate
   ```

   This value allows a the USM to perform pre-validation header manipulation in order of increasing precedence: SIP interface, realm, and session agent.

5. Save and activate the configuration.

Best Practices

This section lists practices that AOracle recommends you follow for successful implementation of this feature.

- Define all storage rules first.
  This recommendation is made because each subsequent header rule processes against the same SIP message, so each additional header rules works off of the results from the application of the rule that precedes it.

  In general, you want to store values from the original SIP header rather than from the iteratively changed versions.
- Implement rules at the element rule rather than the header rule level.
  Header rules should only be a container for element rules.
- When you are creating rules to edit a header, add additional element rules to modify a single header rather than try to create multiple header rules each with one element rule. That is, create multiple element rules within a header rule rather than creating multiple header rules.

- Do not use header or element rule names that are all capital letters (i.e., $IP_ADDRESS). Capitals currently refer to predefined rules that are used as macros, and they might conflict with a name that uses capital letters.

About Regular Expressions

Two of the most fundamental ideas you need to know in order to work with regular expressions and with this feature are:

- Regular expressions are a way of creating strings to match other string values.
- You can use groupings in order to create stored values on which you can then operate.

To learn more about regex, you can visit the following Web site, which has information and tutorials that can help to get you started:http://www.regular-expressions.info/.

Many of the characters you can type on your keyboard are literal, ordinary characters—they present their actual value in the pattern. Some characters have special meaning, however, and they instruct the regex function (or engine which interprets the expressions) to treat the characters in designated ways. The following table outlines these “special characters” or metacharacters.

Character	Name	Description
.	dot	Matches any one character, including a space; it will match one character, but there must be one character to match. Literally a . (dot) when bracketed ([]), or placed next to a \ (backslash).
*	star/asterisk	Matches one or more preceding character (0, 1, or any number), bracketed carrier class, or group in parentheses. Used for quantification. Typically used with a . (dot) in the format .* to indicate that a match for any character, 0 or more times. Literally an * (asterisk) when bracketed ([]).
+	plus	Matches one or more of the preceding character, bracketed carrier class, or group in parentheses. Used for quantification. Literally a + (plus sign) when bracketed ([]).
|	bar/vertical bar/pipe	Matches anything to the left or to the right; the bar separates the alternatives. Both sides are not always tried; if the left does not match, only then is the right attempted. Used for alternation.
{	left brace	Begins an interval range, ended with } (right brace) to match; identifies how many times the previous singles character or group in parentheses must repeat. Interval ranges are entered as minimum and maximums ({minimum,maximum}) where the character/group must appear a minimum of times up to the maximum. You can also use these character to set magnitude, or exactly the number of times a character must appear; you can set this, for example, as the minimum value without the maximum ({minimum,}).

Character	Name	Description
?	question mark	Signifies that the preceding character or group in parentheses is optional; the character or group can appear not at all or one time.
^	caret	Acts as an anchor to represent the beginning of a string.
$	dollar sign	Acts as an anchor to represent the end of a string.
[	left bracket	Acts as the start of a bracketed character class, ended with the] (right bracket). A character class is a list of character options; one and only on of the characters in the bracketed class must appear for a match. A - (dash) in between two character enclosed by brackets designates a range; for example [a-z] is the character range of the lower case twenty-six letters of the alphabet. Note that the] (right bracket) ends a bracketed character class unless it sits directly next to the [(left bracket) or the ^ (caret); in those two cases, it is the literal character.
(	left parenthesis	Creates a grouping when used with the) (right parenthesis). Groupings have two functions: They separate pattern strings so that a whole string can have special characters within it as if it were a single character. They allow the designated pattern to be stored and referenced later (so that other operations can be performed on it).

Expression Building Using Parentheses

You can now use parentheses (())when you use HMR to support order of operations and to simplify header manipulation rules that might otherwise prove complex. This means that expressions such as (sip + urp) - (u + rp) can now be evaluated to sip. Previously, the same expression would have evaluated to sipurprp. In addition, you previously would have been required to create several different manipulation rules to perform the same expression.

SIP Manipulation Configuration

This section explains the parameters that appear in the subelements for the SIP manipulations configuration. Within the SIP manipulations configuration, you can set up SIP header rules, and within those header rules you can configure element rules.

This section also contains several configuration examples for different applications of the HMR feature.

Configuring SIP Header Manipulation Rules

To configure dynamic SIP header manipulation rules:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-manipulation** and press Enter.

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. Type header-rules and press Enter.

   ```
   ORACLE(sip-manipulation)# header-rules
   ```

5. **name**—Enter the unique identifier for this SIP HMR. There is no default for this value.
6. **header-name**—Enter the name of the header on which you want the Oracle USM (USM) to use this HMR. There is no default for this parameter.

   Set this parameter to @status-line, where the at-sign (@)—not allowed in SIP header names—to prevent undesired matches with header having the name status-code.
7. **msg-type**—Specify the type of message to which this SIP HMR will be applied. The default value is **any**. The valid values are:

   - any | request | reply
8. **methods**—Enter the method type to use when this SIP HMR is used, such as INVITE, ACK, or CANCEL. When you do not set the method, the USM applies the rule across all SIP methods.
9. **comparison-type**—Enter the way that you want SIP headers to be compared from one of the available. This choice dictates how the USM processes the match rules against the SIP header. the default is **refer-case-sensitive.** The valid values are:.

   - boolean | refer-case-sensitive | refer-case-insensitive | pattern-rule | case-sensitive | case-insensitive
10. **action**—Enter the action that you want this rule to perform on the SIP header. The default value is **none**. The valid values are:

    - add | delete | manipulate | store | none

      Remember that you should enter rules with the action type store before you enter rules with other types of actions.

      When you set the action type to store, the USM always treats the match value you enter as a regular expression. As a default, the regular expression is uses for the match value is .+ (which indicates a match value of at least one character), unless you set a more specific regular expression match value.
11. **match-value**—Enter the value to match against the header value in SIP packets; the USM matches these against the entire SIP header value. This is where you can enter values to match using regular expression values. Your entries can contain Boolean operators.

    When you configure HMR (using SIP manipulation rules, elements rules, etc.), you can now use escape characters in the **match-value** parameter to support escaping Boolean and string manipulation operators..

    You can also escape the escape character itself, so that it is used as a literal string. For example, the USM now treats the string \+1234 as +1234.

    The following are escape characters: +, -, +^, -^, &, |, \, (,), ., $, ^, and “.

    You can also use two variables, $REMOTE_PORT and $LOCAL_PORT, which resolve respectively to the far-end and remote UDP or TCP port value.

12. **new-value**—When the action parameter is set to add or to manipulate, enter the new value that you want to substitute for the entire header value. This is where you can set stored regular expression values for the USM to use when it adds or manipulates SIP headers.

    When you configure HMR (using SIP manipulation rules, elements rules, etc.), you can now use escape characters in the **new-value** parameter to support escaping Boolean and string manipulation operators..

    You can also escape the escape character itself, so that it is used as a literal string. For example, the USM now treats the string \+1234 as +1234.

    The following are escape characters: +, -, +^, -^, &, |, \, (,), ., $, ^, and “.

    You can also use two variables, $REMOTE_PORT and $LOCAL_PORT, which resolve respectively to the far-end and remote UDP or TCP port value.

Configuring SIP Header Manipulation Element Rules

Element rules are a subset of the SIP header manipulation rules and are applied at the element type level rather than at the entire header value.

To configure dynamic SIP header manipulation rules:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter.

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. Type **header-rules** and press Enter.

   ```
   ORACLE(sip-manipulation)# header-rules
   ```

5. Type **element-rules** and press Enter.

   ```
   ORACLE(sip-header-rules)# element-rules
   ORACLE(sip-element-rules)#
   ```

6. **name**—Enter the unique identifier for this element rule. There is no default for this value.
7. **parameter-name**—Enter the SIP header parameter/element on which you want the Oracle USM (USM) to use this rule. There is no default for this parameter.
8. **type**—Specify the type of parameter to which this element rule will be applied. The default value is **none**. The valid values are:

   - header-value | header-param-name | header-param | uri-display | uri-user | uri-user-param | uri-host | uri-port | uri-param-name | uri-param | uri-header-name | uri-header

     To configure HMR so that there is impact only on the status-line; the value will be used for matching according to the **comparison-type**:

   - **status-code**—Designates the status code of the response line; accepts any string, but during the manipulation process only recognizes the range from 1 to 699.
   - **reason-phrase**—Designates the reason of the response line; accepts any string.

9. **match-val-type**—Enter the value type that you want to match when this rule is applied. The default value is **ANY**. Valid values are:

   - IP | FQDN | ANY

10. **comparison-type**—Enter the way that you want SIP headers to be compared from one of the available. This choice dictates how the USM processes the match rules against the SIP header parameter/element. The default is **refer-case-sensitive**.

    - boolean | refer-case-sensitive | refer-case-insensitive | pattern-rule

11. **action**—Enter the action that you want this rule to perform on the SIP header parameter/element. The default is **none**. The valid rules are:

    - add | replace | delete-element | delete-header | store | none

      Remember that you should enter rules with the action type store before you enter rules with other types of actions.

      When you set the action type to store, the USM always treats the match value you enter as a regular expression. As a default, the regular expression is uses for the match value is .+ (which indicates a match value of at least one character), unless you set a more specific regular expression match value.

12. **match-value**—Enter the value to match against the header value in SIP packets; the USM matches these against the value of the parameter/element. This is where you can enter values to match using regular expression values, or stored pattern matches. Your entries can contain Boolean operators.

    When you configure HMR (using SIP manipulation rules, elements rules, etc.), you can now use escape characters in the **match-value** parameter to support escaping Boolean and string manipulation operators..

    You can also escape the escape character itself, so that it is used as a literal string. For example, the USM now treats the string \+1234 as +1234.

    The following are escape characters: +, -, +^, -^, &, |, \, (,), ., $, ^, and “.

    You can also use two variables, $REMOTE_PORT and $LOCAL_PORT, which resolve respectively to the far-end and remote UDP or TCP port value.

13. **new-value**—When the action parameter is set to add or to manipulate, enter the new value that you want to substitute for the entire header value. This is where you can set stored regular expression values for the USM to use when it adds or manipulates parameters/elements.

    When you configure HMR (using SIP manipulation rules, elements rules, etc.), you can now use escape characters in the **new-value** parameter to support escaping Boolean and string manipulation operators..

    You can also escape the escape character itself, so that it is used as a literal string. For example, the USM now treats the string \+1234 as +1234.

    The following are escape characters: +, -, +^, -^, &, |, \, (,), ., $, ^, and “.

    You can also use two variables, $REMOTE_PORT and $LOCAL_PORT, which resolve respectively to the far-end and remote UDP or TCP port value.

Status-Line Manipulation and Value Matching

The Oracle USM’s HMR feature has been enhanced to support the ability to change the status code or reason phrase in SIP responses. This addition—the ability to edit status-lines in

responses—builds on HMR's existing ability to edit response headers or the Request-URI in a request.

This section shows you how to configure SIP HMR when you want the Oracle USM to drop a 183 Session Progress response when it does not have SDP, though flexibility is built into this feature so that you can use it to achieve other ends. In addition, you can now set the SIP manipulation's **match-value** parameter with Boolean parameters (AND or OR).

Setting the Header Name

SIP header rules (part of the SIP manipulation configuration) now support a new value for the header-name parameter. The value is @status-line, where the at-sign (@)—not allowed in SIP header names—prevents undesired matches with header having the name status-code.

To set the header name for SIP header rules:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter.

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. Type **header-rules** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# header-rules
   ORACLE(sip-header-rules)#
   ```

5. **header-name**—Enter the new value for the **header-name** parameter: @status-line.

Setting the Element Type

In the element rules (a subset of the SIP header rules configuration), you can now set the **type** parameter to either of the following values, both of which will only have an impact on the status-line:

- **status-code**—Designates the status code of the response line; accepts any string, but during the manipulation process only recognizes the range from 1 to 699
- **reason-phrase**—Designates the reason of the response line; accepts any string

Like other rule types you can set, the Oracle USM matches against the value for these using case-sensitive, case-insensitive, or pattern-rule matching (set in the **comparison-type** parameter for the element rule).

To set the element type:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

4. Type **header-rules** and press Enter.

```
ORACLE(session-router)# header-rules
ORACLE(sip-header-rules)#
```

5. Type **element-rule** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(sip-header-rules)# element-rules
ORACLE(sip-element-rules)#
```

6. **type**—Enter either status-code or reason-phrase, the value of which will be used for matching according to the **comparison-type**.

Setting the Match Value

Note that for the SIP header rules and for the SIP element rules, the **match-value** parameter can now be set with these Boolean operators:

- and (for which the syntax is the ampersand &)
- or (for which the syntax is the pipe|)

However, you can only use Boolean operators in this value when you set the **comparison-type** parameter to pattern-rule and are evaluating stored matches. The Oracle USM evaluates these Boolean expressions from left to right, and does not support any grouping mechanisms that might change the order of evaluation. For example, the Oracle USM evaluates the expression A & B | C (where A=true, B=false, and C=true) as follows: A & B = false; false | true = true.

You can set the match-value for the SIP header rules or for the SIP element rules.

To set a match value in the SIP header rules configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

4. Type **header-rules** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# header-rules
ORACLE(sip-header-rules)#
```

5. **match-value**—Enter the value to match against the header value in SIP packets; the Oracle USM matches these against the entire SIP header value. This is where you can enter values to match using regular expression values; your entries can contain Boolean operators.

   To set a match value in the SIP element rules configuration:

6. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

7. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

8. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

9. Type **header-rules** and press Enter.

```
ORACLE(session-router)# header-rules
ORACLE(sip-header-rules)#
```

10. Type **element-rule** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(sip-header-rules)# element-rules
ORACLE(sip-element-rules)#
```

11. **match-value**—Enter the value to match against the header value in SIP packets; the Oracle USM matches these against the value of the parameter/element. This is where you can enter values to match using regular expression values, or stored pattern matches; your entries can contain Boolean operators.

Setting the Response Code Block

To enable the SIP HMR enhancements, you need to set an option in SIP interface configuration that keeps the Oracle USM from sending the response you designate.

Note that this example sets the dropResponse option to 699, where 699 is an arbitrary code used to later match the HMR.

To enable SIP response blocking for a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

If you are adding support for this feature to a pre-existing SIP interface, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **dropResponse** with a plus sign in front of it, type the equal sign and the code(s) or range(s) you want blocked. If there is more than one, separate your entries with a colon. Then press Enter.

```
ORACLE(sip-interface)# options +dropResponse=699
```

If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options

list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Configuring MIME Support

The **find-replace-all** action has been added to the header rules. Element rules support the **find-replace-all** action and the **mime** type.

To set the header rule with the find-replace-all action:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

4. Type **header-rules** and press Enter.

```
ORACLE(sip-manipulation)# header-rules
ORACLE(sip-header-rules)#
```

5. **action**—Set the action parameter to **find-replace-all** if you want to enable SIP HMR MIME manipulation.
6. Save and activate your configuration.

   To set the element rule with the find-replace-all action and MIME type:

7. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

8. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

9. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

10. Type **header-rules** and press Enter.

```
ORACLE(sip-manipulation)# header-rules
ORACLE(sip-header-rules)#
```

11. Type **element-rules** and press Enter.

```
ORACLE(sip-header-rules)# element-rules
```

12. ORACLE(sip-element-rules)#
13. **action**—Set the action parameter to **find-replace-all** if you want to enable SIP HMR MIME manipulation.

14. **type**—Set the type parameter to **mime** if you want to enable SIP HMR MIME manipulation.
15. Save and activate your configuration.

Testing Pattern Rules

The Oracle USM it yields the results you require. This command is useful for testing the regex values that you devise because it will tell you whether that value is valid or not.

This command is called **test-pattern-rule**.

To test a pattern rule:

1. Type **test-pattern-rule** and press Enter.

```
ORACLE# test-pattern-rule
ORACLE(test-pattern-rule)#
```

2. **expression**—Enter the regular expression that you want to test. If there is a match, then the Oracle USM will inform you of it; you will also be informed if there is no match.

   The string against which the Oracle USM is matching is not the string parameter that you can use for this command; it is the string value of the regular expression you entered.

```
ORACLE(test-pattern-rule)# expression '.*;tgid=(.+).*'
```

3. **string**—Enter the string against which you want to compare the regular expression.

```
ORACLE(test-pattern-rule)# string sip:+17024260002@KCMGGWC;user=phone SIP/
2.0;tgid=Trunk1
expression made 3 matches against string
```

4. **show**—Use the **show** command within test-pattern-rules to view the test pattern that you entered, whether there was a match, and the number of matches.

```
ORACLE(test-pattern-rule)# show
Pattern Rule:
Expression : .*(;tgid=(.+)).*
String     : sip:+17024260002@KCMGGWC;user=phone SIP/2.0;tgid=Trunk1
Matched    : TRUE
Matches:
$0 sip:+17024260002@KCMGGWC;user=phone SIP/2.0;tgid=Trunk1
$1 ;tgid=Trunk1
$2 Trunk1
```

Configuring SIP HMR Sets

This section shows you how to configure your multiple SIP HMR sets.

Remember to run the ACLI **verify-config** command prior to activating your configuration so the Oracle USM can detect any invalid or circular references.

To set the parameters enabling the use of SIP HMR sets:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-manipulation** and press Enter.

```
ORACLE(session-router)# sip-manipulation
ORACLE(sip-manipulation)#
```

4. Type **header-rules** and press Enter.

```
ORACLE(session-router)# header-rules
ORACLE(sip-header-rules)#
```

5. Type **element-rule** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(sip-header-rules)# element-rules
ORACLE(sip-element-rules)#
```

6. **action**—Enter the sip-manip value to enable use this rule for a SIP HMR set. This value then invoke the rule identified in the new-value parameter.
7. **new-value**—To use SIP HMR sets, enter the name of the manipulation rule you want invoked for the set.
8. Save and activate your configuration.

Configuration Examples

This section shows you several configuration examples for HMR. This section shows the configuration for the various rules that the Oracle USMapplied, and sample results of the manipulation. These examples present configurations as an entire list of fields and settings for each ruleset, nested header rules and nested element rules. If a field does not have any operation within the set, the field is shown with the setting at the default or blank.

Example 1 Removing Headers

For this manipulation rule, the Oracle USM removes the Custom header if it matches the pattern rule. It stores the defined pattern rule for the goodBye header. Finally, it removes the goodBye header if the pattern rule from above is a match.

This is a sample of the configuration:

```
sip-manipulation
        name                            removeHeader
        header-rule
                name                            removeCustom
                header-name                     Custom
                action                          delete
                comparison-type                 boolean
                match-value                     ^This is my.*
                msg-type                        request
                new-value
                methods                         INVITE
        header-rule
                name                            goodByeHeader
                header-name                     Goodbye
                action                          store
comparison-type     boolean
                match-value                     ^Remove (.+)
                msg-type                        request
                new-value
                methods                         INVITE
header-rule
```

```
                name                            goodBye
action                          delete
                comparison-type                 pattern-rule
                match-value                     $goodByeHeader
                msg-type                        request
                new-value
                methods                         INVITE
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK0g639r10fgc0aakk26s1.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SDc1rm601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SDc1rm601-d01673bcacfcc112c053d95971330335-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
```

Example 2 Manipulating the Request URI

For this manipulation rules, the Oracle USM stores the URI parameter tgid in the Request URI. Then if the pattern rule matches, it adds a new header (x-customer-profile) with the a new header value tgid to the URI parameter in the request URI.

This is a sample of the configuration:

```
sip-manipulation
        name                            CustomerTgid
        header-rule
                name                            ruriRegex
                header-name                     request-uri
                action                          store
                comparison-type                 pattern-rule
                match-value
                msg-type                        request
new-value
                methods                         INVITE
                element-rule
                        name                            tgidParam
                        parameter-name                  tgid
                        type                            uri-param
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value
                        new-value
header-rule
                name                            addCustomer
                header-name                     X-Customer-Profile
                action                          add
                comparison-type                 pattern-rule
                match-value                     $ruriRegex.$tgidParam
```

```
                msg-type                        request
                new-value                       $ruriRegex.$tgidParam.$0
                methods                         INVITE
header-rule
                name                            delTgid
                header-name                     request-uri
                action                          manipulate
                comparison-type                 pattern-rule
                match-value                     $ruriRegex.$tgidParam
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            tgidParam
                        parameter-name                  tgid
                        type                            uri-param
                        action                          delete-element
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value                     $ruriRegex.$tgidParam.$0
                        new-value
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060 SIP/2.0
    Message Header
Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK0g6plv3088h03acgh6c1.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SDc1rg601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SDc1rg601-f125d8b0ec7985c378b04cab9f91cc09-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
        X-Customer-Profile: 123
```

Example 3 Manipulating a Header

For this manipulation rule, the Oracle USMstores the pattern matches for the Custom header, and replaces the value of the Custom header with a combination of the stored matches and new content.

This is a sample of the configuration:

```
sip-manipulation
        name                            modCustomHdr
        header-rule
                name                            customSearch
                header-name                     Custom
                action                          store
                comparison-type                 pattern-rule
                match-value                     (This is my )(.+)( header)
                msg-type                        request
```

```
                new-value
                methods                         INVITE
header-rule
                name                            customMod
                header-name                     Custom
                action                          manipulate
                comparison-type                 pattern-rule
                match-value                     $customSearch
                msg-type                        request
                new-value
methods                         INVITE
                element-rule
                        name                            hdrVal
                        parameter-name                  hdrVal
                        type                            header-value
                        action                          replace
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
new-value                       $customSearch.$1+edited+$customSearch.$3
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK20q2s820boghbacgs6o0.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SDe1ra601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SDe1ra601-4bb668e7ec9eeb92c783c78fd5b26586-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my edited header
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
```

Example 4 Storing and Using URI Parameters

For this manipulation rule, the Oracle USM stores the value of the URI parameter tag from the From header. It also creates a new header FromTag with the header value from the stored information resulting from the first rule.

This is a sample of the configuration:

```
sip-manipulation
        name                            storeElemParam
        header-rule
                name                            Frohmr
                header-name                     From
                action                          store
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
```

```
                element-rule
                        name                            elementRule
                        parameter-name                  tag
                        type                            uri-param
                        action                          store
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value
header-rule
                name                            newHeader
                header-name                     FromTag
                action                          add
                comparison-type                 pattern-rule
                match-value                     $FromHR.$elementRule
                msg-type                        any
                new-value                       $FromHR.$elementRule.$0
                methods
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK4oda2e2050ih7acgh6c1.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SDf1re601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SDf1re601-f85059e74e1b443499587dd2dee504c2-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
Content-Length: 140
        FromTag: 1
```

Example 5 Manipulating Display Names

For this manipulation rule, the Oracle USM sores the display name from the Display header. It replaces the two middle characters of the original display name with a new string. Then is also replaces the From header's display name with "abc 123" if it matches sipp.

This is a sample of the configuration:

```
sip-manipulation
        name                            modDisplayParam
        header-rule
                name                            storeDisplay
                header-name                     Display
                action                          store
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
```

```
                        name                            displayName
                        parameter-name                  display
                        type                            uri-display
                        action                          store
                        match-val-type                  any
comparison-type                 pattern-rule
                        match-value                     (s)(ip)(p )
                        new-value
header-rule
                name                            modDisplay
                header-name                     Display
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            modRule
                        parameter-name                  display
                        type                            uri-display
                        action                          replace
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $storeDisplay.$displayName
                        new-value                       $storeDisplay.
$displayName.$1+lur+$storeDisplay.$displayName.$3
header-rule
                name                            modFrom
                header-name                     From
                action                          manipulate
                comparison-type                 pattern-rule
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            fromDisplay
                        parameter-name
                        type                            uri-display
                        action                          replace
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     sipp
                        new-value                       "\"abc 123\" "
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK681kot109gp04acgs6o0.1
        From: "abc 123" <sip:sipp@192.168.1.60:5060>;tag=SD79ra601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SD79ra601-a487f1259e2370d3dbb558c742d3f8c4-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: slurp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
```

```
Edit: disp <sip:user@192.168.1.60:5060>
Max-Forwards: 69
Subject: Performance Test
Content-Type: application/sdp
Content-Length: 140
```

Example 6 Manipulating Element Parameters

For this more complex manipulation rule, the Oracle USM:

- From the Display header, stores the display name, user name, URI parameter up, and header parameter hp
- Adds the header parameter display to the Params header, with the stored value of the display name from the first step
- Add the URI parameter user to the Params header, with the stored value of the display name from the first step
- If the URI parameter match succeeds in the first step, replaces the URI parameter up with the Display header with the value def
- If the header parameter match succeeds in the first step, deletes the header parameter hp from the Display header

This is a sample of the configuration:

```
sip-manipulation
        name                            elemParams
        header-rule
                name                            StoreDisplay
                header-name                     Display
                action                          store
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            displayName
                        parameter-name
                        type                            uri-display
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value
                        new-value
element-rule
                        name                            userName
                        parameter-name                  user
                        type                            uri-user
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value
                        new-value
element-rule
                        name                            uriParam
                        parameter-name                  up
                        type                            uri-param
                        action                          store
                        match-val-type                  any
```

```
                        comparison-type                 pattern-rule
                        match-value
                        new-value
element-rule
                        name                            headerParam
                        parameter-name                  hp
                        type                            header-param
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value
                        new-value
        header-rule
                name                            EditParams
                header-name                     Params
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            addHeaderParam
                        parameter-name                  display
                        type                            header-param
                        action                          add
match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       $StoreDisplay.
$displayName.$0
                element-rule
                        name                            addUriParam
                        parameter-name                  user
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
               new-value
$StoreDisplay.$userName.$0
        header-rule
                name                            EditDisplay
                header-name                     Display
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            replaceUriParam
                        parameter-name                  up
                        type                            uri-param
                        action                          replace
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $StoreDisplay.$uriParam
                        new-value                       def
                element-rule
                        name                            delHeaderParam
```

```
                        parameter-name                  hp
                        type                            header-param
                        action                          delete-element
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value $StoreDisplay.$headerParam
                        new-value
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK7okvei0028jgdacgh6c1.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SD89rm601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SD89rm601-b5b746cef19d0154cb1f342cb04ec3cb-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: sipp <sip:user@192.168.1.60:5060;up=def>
        Params: sipp <sip:sipp1@192.168.1.60:5060;user=user>;display=sipp
        Params: sipp <sip:sipp2@192.168.1.60:5060;user=user>;display=sipp
        Edit: disp <sip:user@192.168.1.60:5060>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
```

Example 7 Accessing Data from Multiple Headers of the Same Type

For this manipulation rule, the Oracle USM stores the user name from the Params header. It then adds the URI parameter c1 with the value stored from the first Params header. Finally, it adds the URI parameter c2 with the value stored from the second Params header.

This is a sample of the configuration:

```
sip-manipulation
        name                            Params
        header-rule
                name                            storeParams
                header-name                     Params
                action                          store
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            storeUserName
                        parameter-name                  user
                        type                            uri-user
                        action                          store
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value
header-rule
                name                            modEdit
                header-name                     Edit
                action                          manipulate
```

```
                comparison-type                pattern-rule
                match-value
                msg-type                       request
                new-value
methods                          INVITE
                element-rule
                        name                                addParam1
                        parameter-name                      c1
                        type                                uri-param
                        action                              add
                        match-val-type                      any
                        comparison-type                     case-sensitive
                        match-value
                        new-value                           $storeParams[0].
$storeUserName.$0
                element-rule
                        name                                addParam2
                        parameter-name                      c2
                        type                                uri-param
                        action                              add
                        match-val-type                      any
                        comparison-type                     case-sensitive
                        match-value
                        new-value                           $storeParams[1].
$storeUserName.$0
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK9g855p30cos08acgs6o0.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SD99ri601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SD99ri601-6f5691f6461356f607b0737e4039caec-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060;c1=sipp1;c2=sipp2>
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
```

Example 8 Using Header Rule Special Characters

For this manipulation rule, the Oracle USM:

- Stores the header value of the Params header with the given pattern rule, and stores both the user name of the Params header and the URI parameter abc
- Adds the URI parameter lpu with the value stored from the previous Params header
- If any of the Params headers match the pattern rule defined in the first step, adds the URI parameter apu with the value aup
- If all of the Params headers match the pattern rule defined in the first step, adds the URI parameter apu with the value apu

- If the first Params headers does not match the pattern rule for storing the URI parameter defined in the first step, adds the URI parameter not with the value 123
- If the first Params headers matches the pattern rule for storing the URI parameter defined in the first step, adds the URI parameter yes with the value 456

This is a sample of the configuration:

```
sip-manipulation
        name                            specialChar
        header-rule
                name                            searchParams
                header-name                     Params
                action                          store
                comparison-type                 pattern-rule
                match-value                     .*sip:(.+)@.*
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            userName
                        parameter-name
                        type                            uri-user
                        action                          store
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value
element-rule
                        name                            emptyUriParam
                        parameter-name                  abc
                        type                            uri-param
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value
                        new-value
header-rule
                name                            addUserLast
                header-name                     Edit
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            lastParamUser
                        parameter-name                  lpu
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value $searchParams[^].$userName.$0
                element-rule
                        name                            anyParamUser
                        parameter-name                  apu
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 pattern-rule
```

```
                        match-value                     $searchParams[~]
                        new-value                       aup
                element-rule
                        name                            allParamUser
                        parameter-name                  apu
                        type                            header-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $searchParams[*]
                        new-value                       apu
                element-rule
                        name                            notParamYes
                        parameter-name                  not
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     !$searchParams.
$emptyUriParam
                        new-value                       123
                element-rule
                        name                            notParamNo
                        parameter-name                  yes
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $searchParams.
$emptyUriParam
                        new-value                       456
```

This is a sample of the result:

```
Request-Line: INVITE sip:service@192.168.200.60:5060;tgid=123 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP 192.168.200.61:5060;branch=z9hG4bK681m9t30e0qh6akgj2s1.1
        From: sipp <sip:sipp@192.168.1.60:5060>;tag=SDchrc601-1
        To: sut <sip:service@192.168.1.61:5060>
        Call-ID: SDchrc601-fcf5660a56e2131fd27f12fcbd169fe8-06a3gu0
        CSeq: 1 INVITE
        Contact: <sip:sipp@192.168.200.61:5060;transport=udp>
        Goodbye: Remove Me
        Custom: This is my custom header
        Display: sipp <sip:user@192.168.1.60:5060;up=abc>;hp=123
        Params: sipp <sip:sipp1@192.168.1.60:5060>
        Params: sipp <sip:sipp2@192.168.1.60:5060>
        Edit: disp <sip:user@192.168.1.60:5060;lpu=sipp2;apu=aup;not=123>;apu=apu
        Max-Forwards: 69
        Subject: Performance Test
        Content-Type: application/sdp
        Content-Length: 140
```

Example 9 Status-Line Manipulation

This section shows an HMR configuration set up for status-line manipulation.

Given that the object of this example is to drop the 183 Session Progress response when it does not have SDP, your SIP manipulation configuration needs to:

1. Search for the 183 Session Progress response

2. Determine if the identified 183 Session Progress responses contain SDP; the Oracle USM searches the 183 Session Progress responses where the content length is zero
3. If the 183 Session Progress response does not contain SDP, change its status code to 699
4. Drop all 699 responses

```
sip-manipulation
        name                            manip
        description
        header-rule
                name                            IsContentLength0
                header-name                     Content-Length
                action                          store
                comparison-type                 pattern-rule
                match-value                     0
                msg-type                        reply
                new-value
                methods
        header-rule
                name                            is183
                header-name                     @status-line
                action                          store
                comparison-type                 pattern-rule
                match-value
                msg-type                        reply
                new-value
                methods
                element-rule
name                            is183Code
                        parameter-name
                        type                            status-code
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     183
                        new-value
        header-rule
                name                            change183
                header-name                     @status-line
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        reply
                new-value
                methods
                element-rule
                        name                            make199
                        parameter-name
                        type                            status-code
                        action                          replace
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $IsContentLength0 &
$is183.$is183Code
                        new-value                       199

sip-interface     options dropResponse=699
```

Example 10 Use of SIP HMR Sets

The following example shows the configuration for SIP HMR with one SIP manipulation configuration loading another SIP manipulation configuration. The goals of this configuration are to:

- Add a new header to an INVITE
- Store the user portion of the Request URI
- Remove all Route headers from the message only if the Request URI is from a specific user

```
sip-manipulation
        name                            deleteRoute
        description                     delete all Route Headers
        header-rule
                name                            deleteRoute
                header-name                     Route
                action                          delete
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
sip-manipulation
        name                            addAndDelete
        description                     Add a New header and delete Route
headers
        header-rule
                name                            addHeader
                header-name                     New
                action                          add
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value                       "Some Value"
                methods                         INVITE
        header-rule
                name                            storeRURI
                header-name                     request-uri
                action                          store
                comparison-type                 pattern-rule
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            storeUser
                        parameter-name
                        type                            uri-user
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     305.*
                        new-value
        header-rule
                name                            deleteHeader
                header-name                     request-uri
                action                          sip-manip
```

```
                comparison-type                 Boolean
                match-value                     $storeRURI.$storeUser
                msg-type                        request
                new-value                       deleteRoute
                methods                         INVITE
```

Example 11 Use of Remote and Local Port Information

The following example shows the configuration for remote and local port information. The goals of this configuration are to:

- Add LOCAL_PORT as a header parameter to the From header
- Add REMOTE_PORT as a header parameter to the From header

```
sip-manipulation
        name                            addOrigIp
        description
        header-rule
                name                            addIpParam
                header-name                     From
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            addIpParam
                        parameter-name                  newParam
                        type                            header-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       $LOCAL_IP
                element-rule
                        name                            addLocalPort
                        parameter-name                  lport
                        type                            header-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       $LOCAL_PORT
                element-rule
                        name                            addRemotePort
                        parameter-name                  rport
                        type                            header-param
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       $REMOTE_PORT
```

Example 12 Response Status Processing

Given that the object of this example is to drop the 183 Session Progress response when it does not have SDP, your SIP manipulation configuration needs to:

1. Search for the 183 Session Progress response
2. Determine if the identified 183 Session Progress responses contain SDP; the Oracle USM searches the 183 Session Progress responses where the content length is zero
3. If the 183 Session Progress response does not contain SDP, change its status code to 699
4. Drop all 699 responses

```
sip-manipulation
        name                            manip
        description
        header-rule
                name                            IsContentLength0
                header-name                     Content-Length
                action                          store
                comparison-type                 pattern-rule
                match-value                     0
                msg-type                        reply
                new-value
                methods
        header-rule
                name                            is183
                header-name                     @status-line
                action                          store
                comparison-type                 pattern-rule
                match-value
                msg-type                        reply
                new-value
                methods
                element-rule
                        name                            is183Code
                        parameter-name
                        type                            status-code
                        action                          store
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     183
                        new-value
        header-rule
                name                            change183
                header-name                     @status-line
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        reply
                new-value
                methods
                element-rule
                        name                            make699
                        parameter-name
                        type                            status-code
                        action                          replace
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     $IsContentLength0 &
$is183.$is183Code
                        new-value                       699
sip-interface
    options dropResponse=699
```

The following four configuration examples are based on the this sample SIP INVITE:

```
INVITE sip:service@192.168.1.61:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.60:5060;branch=z9hG4bK-1-0
From: sipp <sip:sipp@192.168.1.60:5060>;tag=1
To: sut <sip:service@192.168.1.61:5060>
Call-ID: 1-15554@192.168.1.60
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.1.60:5060;user=phone>
Max-Forwards: 70
Content-Type: multipart/mixed;boundary=boundary
Content-Length:  466
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=audio 12345 RTP/AVP 18
a=rtpmap:8 G729/8000/1
a=fmtp:18 annexb=no
a=sendrecv
a=ptime:20
a=maxptime:200
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=video 12345 RTP/AVP 34
a=rtpmap:34 H263a/90000
a=ptime:30
--boundary--
```

Example 13 Remove a Line from SDP

In this example, the SIP manipulation is configured to remove all p-time attributes from the SDP.

```
sip-manipulation
        name                            removePtimeFromBody
        description                   removes ptime attribute from all bodies
        header-rule
                name                            CTypeManp
                header-name                     Content-Type
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            remPtime
                        parameter-name                  application/sdp
                        type                            mime
                        action                          find-replace-all
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value                     a=ptime:[0-9]{1,2}(\n|\r
```

```
\n)
                        new-value
```

The result of manipulating the original SIP INVITE (shown above) with the configured SIP manipulation is:

```
INVITE sip:service@192.168.1.61:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.60:5060;branch=z9hG4bK-1-0
From: sipp <sip:sipp@192.168.1.60:5060>;tag=1
To: sut <sip:service@192.168.1.61:5060>
Call-ID: 1-15554@192.168.1.60
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.1.60:5060;user=phone>
Max-Forwards: 70
Content-Type: multipart/mixed;boundary=boundary
Content-Length:  466
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=audio 12345 RTP/AVP 18
a=rtpmap:18 G729/8000/1
a=fmtp:18 annexb=no
a=sendrecv
a=maxptime:200
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=video 12345 RTP/AVP 34
a=rtpmap:34 H263a/90000
--boundary-
```

Example 14 Back Reference Syntax

In this sample of back-reference syntax use, the goal is to change the To user. The SIP manipulation would be configured like the following:

```
sip-manipulation
        name                            changeToUser
        description                   change user in the To header
        header-rule
                name                            ChangeHeader
                header-name                     To
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        request
                new-value
                methods                         INVITE
                element-rule
                        name                            replaceValue
                        parameter-name
                        type                            header-value
                        action                          replace
```

```
                        match-val-type                  any
                        comparison-type                 pattern-rule
                        match-value                     (.+)(service)(.+)
                        new-value                       $1+Bob+$3
```

The result of manipulating the original SIP INVITE (shown above) with the configured SIP manipulation is:

```
INVITE sip:service@192.168.1.61:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.60:5060;branch=z9hG4bK-1-0
From: sipp <sip:sipp@192.168.1.60:5060>;tag=1
To: sut <sip:Bob@192.168.1.61:5060>
Call-ID: 1-15554@192.168.1.60
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.1.60:5060;user=phone>
Max-Forwards: 70
Content-Type: multipart/mixed;boundary=boundary
Content-Length:  466
…
…
…
```

Example 15 Change and Remove Lines from SDP

In this sample of changing and removing lines from the SDP, the goal is to convert the G.729 codec to G.729a. The SIP manipulation would be configured like the following:

```
sip-manipulation
        name                            std2prop-codec-name
        description                     rule to translate standard to proprietary
codec name
        header-rule
                name                            CTypeManp
                header-name                     Content-Type
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        any
                new-value
                methods
                element-rule
                        name                            g729-annexb-no-std2prop
                        parameter-name                  application/sdp
                        type                            mime
                        action                          find-replace-all
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value                     a=rtpmap:[0-9]{1,3}
(G729/8000/1\r\na=fmtp:[0-9]{1,3} annexb=no)[[:1:]]
                        new-value                       G729a/8000/1
```

The result of manipulating the original SIP INVITE (shown above) with the configured SIP manipulation is:

```
INVITE sip:service@192.168.1.61:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.60:5060;branch=z9hG4bK-1-0
From: sipp <sip:sipp@192.168.1.60:5060>;tag=1
To: sut <sip:service@192.168.1.61:5060>
Call-ID: 1-15554@192.168.1.60
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.1.60:5060;user=phone>
```

```
Max-Forwards: 70
Content-Type: multipart/mixed;boundary=boundary
Content-Length:  466
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=audio 12345 RTP/AVP 8
a=rtpmap:18 G729a/8000/1
a=sendrecv
a=maxptime:200
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=video 12345 RTP/AVP 34
a=rtpmap:34 H263a/90000
--boundary-
```

Example 16 Change and Add New Lines to the SDP

In this sample of changing and adding lines from the SDP, the goal is to convert non-standard codec H.263a to H.263. The SIP manipulation would be configured like the following:

```
sip-manipulation
        name                            prop2std-codec-name
        description                     rule to translate proprietary to standard
codec name
        header-rule
                name                            CodecManp
                header-name                     Content-Type
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                msg-type                        any
                new-value
                methods
                element-rule
                        name                            H263a-prop2std
                        parameter-name                  application/sdp
                        type                            mime
                        action                          find-replace-all
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value                     a=rtpmap:([0-9]{1,3})
H263a/.*\r\n
                        new-value                       a=rtpmap:+$1+"
H263/90000"+$CRLF+a=fmtp:+$1+" QCIF=4"+$CRLF
```

The result of manipulating the original SIP INVITE (shown above) with the configured SIP manipulation is:

```
INVITE sip:service@192.168.1.61:5060 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.60:5060;branch=z9hG4bK-1-0
From: sipp <sip:sipp@192.168.1.60:5060>;tag=1
```

```
To: sut <sip:service@192.168.1.61:5060>
Call-ID: 1-15554@192.168.1.60
CSeq: 1 INVITE
Contact: <sip:sipp@192.168.1.60:5060;user=phone>
Max-Forwards: 70
Content-Type: multipart/mixed;boundary=boundary
Content-Length:  466
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=audio 12345 RTP/AVP 8
a=rtpmap:18 G729/8000/1
a=fmtp:18 annexb=no
a=sendrecv
a=maxptime:200
--boundary
Content-Type: application/sdp
v=0
o=user1 53655765 2353687637 IN IP4 192.168.1.60
s=-
c=IN IP4 192.168.1.60
t=0 0
m=video 12345 RTP/AVP 34
a=rtpmap:34 H263/90000
a=fmtp:34 QCIF=4
--boundary-
```

Dialog-Matching Header Manipulation

The most common headers to manipulate using HMR are the To-URI and From-URI. Along with the to-tag, from-tag, and Call-ID values, these are also all headers that represent dialog-specific information that must match the UAC and UAS to be considered part of the same dialog. If these parameters are modified through HMR, the results can be that the UAC or UAS rejects messages.

While it is possible to ensure that dialog parameters match correctly using regular HMR, this feature offers a simpler and less error-prone method of doing so.

In addition, this section describes the addition of built-in SIP manipulations defined by Oracle best practices, and a new method of testing your SIP manipulations.

About Dialog-Matching Header Manipulations

The goal of this feature is to maintain proper dialog-matching through manipulation of dialog-specific information using HMR. Two fundamental challenges arise when looking at the issue of correctly parameters manipulating dialog-matching:

- Inbound HMR
- Outbound HMR

The new setting **out-of-dialog** (for the **msg-type** parameter) addresses these challenges by offering an intelligent more of dialog matching of messages for inbound and outbound HMR requests. This is a msg-type parameter, meaning that it becomes matching criteria for

operations performed against a message. If you also specify methods (such as REGISTER) as matching criteria, then the rule is further limited to the designated method.

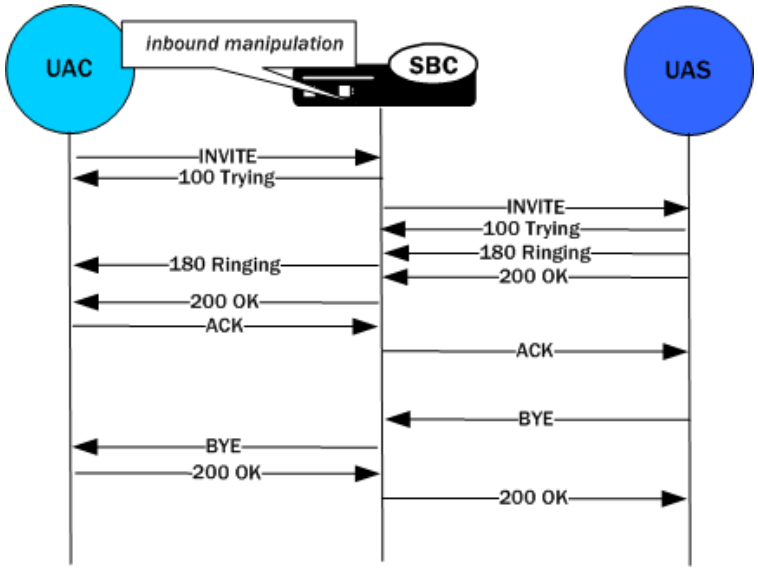
For both inbound and outbound manipulations, using the **out-of-dialog** setting means the message must be a request without a to-tag in order to perform the manipulation.

Inbound HMR Challenge

Since inbound manipulations take place before the message reaches the core of Oracle USM SIP processing, the SIP proxy takes the manipulated header as what was directly received from the client. This can cause problems for requests leaving the Oracle USM for the UAC because the dialog will not match the initial request sent.

So the unmodified header must be cached because for any subsequent request (as in the case of a BYE originating from the terminator; see the diagram below) the Oracle USM might need to restore the original value—enabling the UAC to identify the message correctly as being part of the same dialog. For out-of-dialog requests (when the To, From, or Call-ID headers are modified) the original header will be stored in the dialog when the **msg-type out-of-dialog** is used.

The Oracle USM performs the restoration of original headers outside of SIP manipulations. That is, there are no manipulation rules to configure for restore the header to their original context. The Oracle USM will recognize the headers have been modified, and restore them to their original state prior to sending the message out on the wire. Restoration takes place prior to outbound manipulations so that any outbound manipulation can those headers once they have been restored.

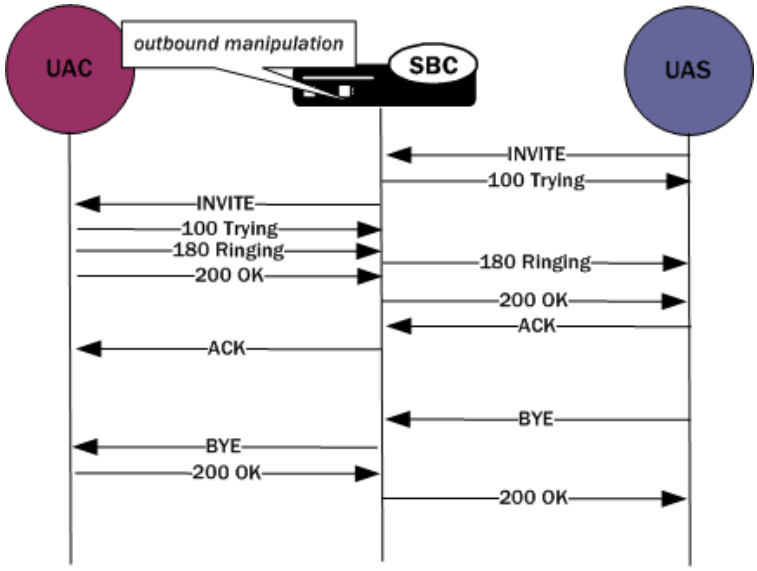


Outbound HMR Challenge

When you use the **out-of-dialog** setting for an outbound manipulation, the Oracle USM only executes this specific SIP header rule only if the same SIP header rule was executed against the initial dialog-creating request.

For example, if the INVITE's To header was not manipulated, it would not be correct to manipulate the To header in the BYE request. To do so would render the UAC unable to properly match the dialog. And this also means that the outbound manipulation should be

carried out against a To, From, or Call-ID header in the BYE request if it was manipulated in the INVITE.



Dialog-matching Header Manipulation Configuration

You using the **out-of-dialog** setting in the **msg-type** parameter, part of the SIP header rules configuration.

To enable dialog-matching header manipulation:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter.

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. **Type mime-rules and press Enter. If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE(sip-manipulation)# header-rules
   ORACLE(sip-header-rules)#
   ```

5. **msg-type**—Set this parameter to **out-of-dialog** to enable dialog-matching header manipulation.

6. Save your work.

Built-In SIP Manipulations

In the course of HMR use, certain rules have become commonly used. Lengthy and complex, these rules do not include any customer-specific information and do they can be used widely.

To make using them easier, they have been turned into built-in rules that you can reference in the **in-manipulationid** and **out-manipulationid** parameters that are part of the realm, session agent, and SIP interfaces configurations.

Built-in rules start with the prefix ACME_, so Oracle recommends you name your own rules in a different manner to avoid conflict.

While the number of built-in manipulation rules is expected to grow, one is supported at the present time: ACME_NAT_TO_FROM_IP. When performed outbound, this rule changes:

- The To-URI hostname to the logical $TARGET_IP and port to $TARGET_PORT
- The From-URI to the logical $REPLY_IP and port to be $REPLY_PORT

Built-In SIP Manipulation Configuration

When you want to enable this feature for a realm, session agent, or SIP interface, you configure the **in-manipulationid or out-manipulationid** parameters with the rule.

The sample here shows this feature being applied to a session agent, but the realm and SIP interface configurations also have the same parameter you use to set up the feature.

To use built-in SIP manipulations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **out-manipulationid**—Enter name of the built-in rule you want to use. Remember that all built-in rules start with ACME_.
5. Save your work.

Testing SIP Manipulations

You can now use a new tool that allows you to test the outcome of your SIP manipulation and header rules without sending real traffic through the Oracle USM to see if they work.

To use the tool, you enter the ACLI's test-sip-manipulation utility and reference the rule you want to test using this name. Then you enter a mode where you put in a SIP message entered in ASCII. You can cut and paste this message from sipmsg.log or from some other location. Using <Ctrl-D> stops the SIP message collection and parses it.

The test informs you of any parsing errors found in the SIP message. Once the message is entered, you can execute the SIP manipulation against the message. The output after this step is the modified SIP message after manipulations have been applied. You will also find a debugging option, which displays SIP manipulation logging to the screen as the manipulation takes place.

As a starting point for testing, this tool comes loaded with a default SIP message. It cannot be associated with realms, session agents, or SIP interfaces, and so it also comes with certain

resolves reserved words, such as: $LOCAL_IP, $TRUNK_GROUP_CONTEXT, and $REMOTE_PORT. In addition, you can use your settings for testing across terminal sessions; if you choose to save your settings, everything (including the SIP message) will be saved, with the exception of the debugging option.

It is not recommended that you use this tool to add an ISUP message body.

HMR Import-Export

Due to the complexity of SIP manipulations rules and the deep understanding of system syntax they require, it is often difficult to configure reliable rules. This feature provides support for importing and exporting pieces of SIP manipulation configuration in a reliable way so that they can be reused.

Exporting

The SIP manipulation configuration contains an **export** command. When you use it, the Oracle USM sends the configuration you have selected to a designated file. The contents are the same information you see when you use the ACLI **show** command in XML format; it includes the selected configuration and any changes that have been made. Because you can only export one SIP manipulation configuration at a time, you must export each one-by-one if you need more than one.

The file name can be any you selected, and would be most useful if it were to identify its contents in some way. If the file already exists, then the export fails and informs you the file already exists. A successfully-executed export simply returns you to the system prompt.

The system writes exported files to /code/imports, a new location that will be created to avoid overlap with existing backup files. The files will carry the extension .gz to show that they have been compressed with gzip.

Your export data will look like this sample:

```
<?xml version='1.0' standalone='yes'?>
<sipManipulation
        name='manip'
        description=''
        lastModifiedBy='admin@console'
        lastModifiedDate='2009-10-16 14:16:29'>
        <headerRule
                headerName='Foo'
                msgType='any'
                name='headerRule'
                action='manipulate'
                cmpType='boolean'
                matchValue='$REGEX("[bB][A-Za-z]{2}")'
                newValue='foo'
                methods='INVITE'>
        </headerRule>
</sipManipulation>
```

To avoid conflict with other objects on the system, key and object ID are not included as part of the exported XML.

Importing

Using the import command in the SIP manipulation configuration, you can import data from an exported file to a currently-selected configuration. If you have not selected a configuration into which to load the data, a new one will be created. Including the .gz extension, you enter the full name of the file you want imported. After it finds the file, the Oracle USM unarchives it and parses its contents. If these steps fail, the Oracle USM will alert you. If they succeed, then the configuration data loads into the object.

If you have been making changes to the configuration into which data was imported, the Oracle USM will inform you prior to importing the data so that you will not lose any of your work. This way, you will be less likely to overwrite unsaved changes.

Once the import is complete, it will be as if you entered the configuration by hand. You only need to save your work (by typing **done**) to save the SIP manipulation to the global SIP configuration. Note that if for some reason the XML is malformed or contained more than one object, the import will fail.

If you attempt to import a configuration with the same key as one that already exists, the system returns an error and prevents you from saving the imported object. In this case, you can delete the object with the same key and then carry out your import, or you can select the object with the same key and perform an import that will overwrite it with new data.

Displaying Imports

You can display imported SIP manipulations data at the system prompt. The command lists all files in the exported files directory, and also tells you if there are none.

Using FTP to Move Files

You can also place exported SIP manipulation configuration files on the Oracle USM using FTP. You need to use the same /code/imports directory to do so.

Removing Files

Using the **delete-import** command with the name of the file you want to delete removes it from the system. Using this command, you can delete files that are no longer useful to you. Carrying out this command is final and there is no warning before you go ahead with the deletion. A failed deletion (for instance, because there is no such file) will produce an error message; a successful deletion simply returns you to the system prompt.

Unique HMR Regex Patterns and Other Changes

In addition to the HMR support it offers, the Oracle USM can now be provisioned with unique regex patterns for each logical remote entity. This supplement to pre-existing HMR functionality saves you provisioning time and saves Oracle USM resources in instances when it was previously necessary to define a unique SIP manipulation per PBX for a small number of customer-specific rules.

Manipulation Pattern Per Remote Entity

On the Oracle USM, you can configure logical remote entities (session agents, realms, and SIP interfaces) with a manipulation pattern string that the system uses as a regular expression. Then the SIP manipulation references this regular expression using the reserved word $MANIP_PATTERN. At runtime, the Oracle USM looks for the logical entity configured with a manipulation pattern string in this order of preference: session agent, realm, and finally SIP interface.

On finding the logical entity configured with the manipulation string, the Oracle USM dynamically determines the expression. When there is an invalid reference to a manipulation pattern, the pattern-rule expression that results will turn out to be the default expression (which is \,+).

When the $MANIP_PATTERN is used in a manipulation rule's **new-value** parameter, it resolves to an empty string, equivalent of no value. Even though this process ends with no value, it still consumes system resources. And so Oraclerecommends you do not use $MANIP_PATTERN as a **new-value** value.

In the following example, the SIP manipulation references the regular expression from a realm configuration:

```
realm-config
        identifier                              net200
        description
        addr-prefix                             0.0.0.0
        network-interfaces                      public:0
        ...
        manipulation-pattern                    Lorem(.+)
sip-manipulation
        name                            manip
        description
        header-rules
                name                            headerRule
                header-name                     Subject
                action                          manipulate
                match-value                     $MANIP_PATTERN
                msg-type                        request
                comparison-type                 pattern-rule
                new-value                       Math
                methods                         INVITE
```

Reject Action

When you use this action type and a condition matching the manipulation rule arises, the Oracle USM rejects the request (though does not drop responses) and increments a counter.

- If the **msg-type** parameter is set to **any** and the message is a response, the Oracle USM increments a counter to show the intention to reject the message—but the message will continue to be processed.
- If the **msg-type** parameter is set to **any** and the message is a request, the Oracle USM performs the rejection and increments the counter.

The **new-value** parameter is designed to supply the status code and reason phrase corresponding to the reject. You can use the following syntax to supply this information: status-code[:reason-phrase]. You do not have to supply the status code and reason phrase information; by default, the system uses 400:Bad Request.

If you do supply this information, then the status code must be a positive integer between 300 and 699. The Oracle USM then provides the reason phrase corresponding to the status code. And if there is no reason phrase, the system uses the one for the applicable reason class.

You can also customize a reason phrase. To do so, you enter the status code followed by a colon (:), being sure to enclose the entire entry in quotation marks () if your reason code includes spaces.

When the Oracle USM performs the **reject** action, the current SIP manipulation stops processing and does not act on any of the rules following the **reject** rule. This course of action is true for nested SIP manipulations that might have been constructed using the **sip-manip** action type.

Reject Action Configuration

To support the **reject** action, two parameters in the **session-router-config** allow you to set how many messages in a certain amount of time cause the Oracle USM to generate an SNMP trap.

To set the reject message number and time window:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-router** and press Enter.

   ```
   ORACLE(session-router)# session-router
   ORACLE(session-router-config)#
   ```

4. **reject-message-threshold**—Enter the minimum number of message rejections allowed in the **reject-message-window** time on the Oracle USM (when using the SIP manipulation action reject) before generating an SNMP trap. The default is 0, meaning this feature is disabled and no trap will be sent.

5. **reject-message-window**—Enter the time in seconds that defines the window for maximum message rejections allowed before generating an SNMP trap.

6. Save your work.

About Counters

The Oracle USM tracks messages that have been flagged for rejection using the **reject** action type. In the **show sipd** display, refer to the Rejected Messages category; there is no distinction between requests and responses.

```
ORACLE# show sipd
13:59:07-102
SIP Status                -- Period -- -------- Lifetime --------
                  Active     High   Total      Total  PerMax    High
Sessions               0        0       0          0       0       0
Subscriptions          0        0       0          0       0       0
Dialogs                0        0       0          0       0       0
CallID Map             0        0       0          0       0       0
Rejections             -        -       0          0       0
ReINVITEs              -        -       0          0       0
Media Sessions         0        0       0          0       0       0
```

```
Media Pending        0       0       0          0       0       0
Client Trans         0       0       0          0       0       0
Server Trans         0       0       0          0       0       0
Resp Contexts        0       0       0          0       0       0
Saved Contexts       0       0       0          0       0       0
Sockets              0       0       0          0       0       0
Req Dropped          -       -       0          0       0
DNS Trans            0       0       0          0       0       0
DNS Sockets          0       0       0          0       0       0
DNS Results          0       0       0          0       0       0
Rejected Msgs        0       0       0          0       0       0
Session Rate = 0.0
Load Rate = 0.0
Remaining Connections = 20000 (max 20000)
```

SNMP Support

The Oracle USM provides SNMP support for the Rejected Messages data, so you can access this information externally. The new MIB objects are:

```
apSysRejectedMessages    OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
                "Number of messages rejected by the SD due to matching criteria."
        ::= { apSysMgmtMIBGeneralObjects 18 }
apSysMgmtRejectedMesagesThresholdExeededTrap        NOTIFICATION-TYPE
        OBJECTS         { apSysRejectedMessages }
        STATUS          current
        DESCRIPTION
        " The trap will be generated when the number of rejected messages exceed
the configured threshold within the configured window."
        ::= { apSystemManagementMonitors 57 }
apSysMgmtRejectedMessagesGroup  OBJECT-GROUP
        OBJECTS {
            apSysRejectedMessages
        }
        STATUS          current
        DESCRIPTION
                "Objects to track the number of messages rejected by the SD."
        ::= {  apSystemManagementGroups 18 }
apSysMgmtRejectedMessagesNotificationsGroup NOTIFICATION-GROUP
      NOTIFICATIONS {
                        apSysMgmtRejectedMesagesThresholdExeededTrap
                      }
                STATUS          current
                DESCRIPTION
                "Traps used for notification of rejected messages"
      ::= {  apSystemManagementNotificationsGroups 26 }
apSmgmtRejectedMessagesCap
            AGENT-CAPABILITIES
            PRODUCT-RELEASE     "Acme Packet SD"
            STATUS               current
            DESCRIPTION         "Acme Packet Agent Capability for enterprise
                                 system management MIB."
            SUPPORTS            APSYSMGMT-MIB
                INCLUDES        {
                                  apSysMgmtRejectedMessagesGroup,
                                  apSysMgmtRejectedMessagesNotificationsGroup
```

```
                                        }
::= { apSmgmtMibCapabilities 37 }
```

Log Action

When you use this action type and a condition matching the manipulation rule arises, the Oracle USM logs information about the current message to a separate log file. This log files will be located on the same core in which the SIP manipulation occurred. On the core where sipt runs, a logfile called matched.log will appear when this action type is executed.

The matched.log file contains a timestamp, received and sent Oracle USM network interface, sent or received IP address:port information, and the peer IP address:port information. It also specifies the rule that triggered the log action in this syntax: rule-type[rule:name]. The request URI, Contact header, To Header, and From header are also present.

```
---------------------------------------
Apr 17 14:17:54.526 On [0:0]192.168.1.84:5060 sent to 192.168.1.60:5060
element-rule[checkRURIPort]
INVITE sip:service@192.168.1.84:5060 SIP/2.0
From: sipp <sip:+2125551212@192.168.1.60:5060>;tag=3035SIPpTag001
To: sut <sip:service@192.168.1.84>
Contact: sip:sipp@192.168.1.60:5060
```

Changes to Storing Pattern Rule Values

Release S-C6.2.0 introduces changes to the framework for storing regular expression results within manipulation rules, altering the way the **store** action works. These changes are beneficial to performance.

In previous releases, when the **store** action is used, the Oracle USM stores all values matching the regular expression defined in the **match-value** parameter for all headers. At runtime, the system evaluates all stored values to find the correct index.

Now, you no longer need to specify the **store** action. The simple fact of referencing another rule tells the system it must store a value. When SIP manipulation is used, the system first checks to see if any values require storing. The **add** action is an exception to this process; storing happens after a header is added.

When referring to a rule, that rule still needs to have a regular expression defined in the match-vale and the comparison type set to pattern-rule; else the default expression will be used.

Removal of Restrictions

The following restrictions related to HMR have been removed in Release S-C6.2.0:

- The action **find-replace-all** now executes all element rules. Previously, no child rules were executed.
- The action **sip-manip** now executes existing all element rules. Previously, no child rules were executed.
- The action **store** now executes existing all element rules. Previously, only child rules with the **store** action were executed.
- The action **add** now executes existing all element rules. Previously, only child rules with the **add** action were executed.

Name Restrictions for Manipulation Rules

Historically, you have been allowed to configure any value for the name parameter within a manipulation rule. This method of naming caused confusion when referencing rules, so now manipulation rules name must follow a specific syntax. They must match the expression ^[[alpha:]][[:alnum:]_]+$ and contain at least one lower case letter.

In other words, the name must:

- Start with a letter, and then it can contain any number of letters, numbers, or underscores
- Contain at least one lower case letter

All pre-existing configurations will continue to function normally. If you want to change a manipulation rule, however, you are required to change its name if it does not follow the new format.

The ACLI **verify-config** command warns you if the system has loaded a configuration containing illegal naming syntax.

Please note that the software allows you to make changes to HMRs, including configuring new functionality to existing rules, as long as you do not change the rule name. This results in an important consideration surrounding HMRs with hyphens in previously configured rule names.

- You can reference stored values in new value names. (Recall that stored values may be rule names.)
- You can perform subtraction in new value names.

If you use a rule names with hyphens within the REGEX of new value names, the system cannot determine whether the hyphen is part of the rule name or is intended to invoke subtraction within the REGEX. For this reason, you need to use great care with legacy HMR naming that includes hyphens.

As a general rule, create new rule names that follow the new rule naming guidelines if you intend to use new functionality in those rules.

New Value Restrictions

To simplify configuration and remove possible ambiguity, the use of boolean and equality operators (==, <=, <, etc.) for **new-value** parameter values has been banned. Since there was no specific functionality tied to their use, their ceasing to be use will have no impact to normal SIP manipulation operations.

Header Manipulation Rules for SDP

The Oracle USM supports SIP header and parameter manipulation rules for four types of SIP message contents:

- headers
- elements within headers
- ASCII-encoded Multipurpose Internet Mail Extensions (MIME) bodies
- binary-encoded MIME ISDN User Part (ISUP) bodies

While Session Description Protocol (SDP) offers and answers can be manipulated in a fashion similar to ASCII-encoded MIME, such manipulation is primitive in that it lacks the ability to operate at the SDP session- and media-levels.

In addition, the system supports a variant of Header Manipulation Rules (HMR) operating on ASCII-encoded SDP bodies, with specific element types for descriptors at both the session-level and media-level, and the ability to apply similar logic to SDP message parts as is done for SIP header elements.

The configuration object, mime-sdp-rules, under sip-manipulation specifically addresses the manipulation of SDP parts in SIP messages. Just as existing header-rules are used to manipulate specific headers of a SIP message, mime-sdp-rules will be used to manipulate the SDP specific mime-attachment of a SIP message.

Platform Support

HMR for SDP is available on the following platforms: Acme Packet 38x0, Acme Packet 4500 and Acme Packet 4600.

SDP Manipulation

mime-sdp-rules function in a similar fashion as header-rules. They provide

- parameters used to match against specific SIP methods and/or message types
- parameters used to match and manipulate all or specified parts of an SDP offer or answer
- a means of comparing search strings or expressions against the entire SDP
- different action types to allow varying forms of manipulation

Since only a single SDP can exist within a SIP message, users need not specify a content-type parameter as is necessary for a mime-rule. A mime-sdp-rule operates on the single SDP within the SIP message. If no SDP exists with the message, one can be added. If the message already contains a mime attachment, adding SDP results in a multipart message.

All manipulations performed against all or parts of the SDP are treated as UTF-8 ASCII encoded text. At the parent-level (mime-sdp-rule) the **match-value** and **new-value** parameters execute against the entire SDP as a single string.

An add action only succeeds in the absence of SDP because a message is allowed only a single SDP offer or answer. A delete operation at the mime-sdp-rule level will remove the SDP entirely.

Note that on an inbound sip-manipulation, SDP manipulations interact with the Oracle USM codec-policy. SDP manipulations also interact with codec reordering and media setup. It is very possible to make changes to the SDP such that the call can not be setup due to invalid media parameters, or settings that will affect the ability to transcode the call. Consequently, user manipulation of the SDP can prove risky, and should be approached with appropriate caution.

Three configuration-objects, sdp-session-rule, sdp-media-rule, and mime-header-rule, exist under the mime-sdp-rule. These objects provide finer grained control of manipulating parts of the SDP.

sdp-session-rule

An sdp-session-rule groups all SDP descriptors, up until the first media line, into a single entity, thus allowing the user to perform manipulation operations on a session-specific portion of the SDP.

Like the mime-sdp-rule, all match-value and new-value operations performed at this level are executed against the entire session group as a complete string. Given the sample SDP below, if an sdp-session-rule is configured, the match-value and new-values operate only on the designated portion.

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

Nested under the sdp-session-rule configuration object is an sdp-line-rule object, the object that identifies individual descriptors within the SDP. The types of descriptors used at the sdp-session-rule level are v, o, s, i, u, e, p, c, b, t, r, z, k, and a, the descriptors specific to the entire session description.

This level of granularity affords the user a very simple way to making subtle changes to the session portion of the SDP. For instance, it is very common to have to change the connection line at the session level.

The add and delete actions perform no operation at the sdp-session-rule level.

sdp-media-rule

An sdp-media-rule groups all of the descriptors that are associated with a specific media-type into single entity, thus allowing the user to perform manipulation operations on a media-specific portion of the SDP. For example, a user can construct an sdp-media-rule to change an attribute of the audio media type.

Like a mime-sdp-rule, all match-value and new-value operations performed at this level are executed against the entire media-group as a complete string. Given the sample SDP below, if a media-level-descriptor is configured to operate against the application group, the match-value and new-values would operate only on designated portion.

```
        v=0
        o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
        s=SDP Seminar
        i=A Seminar on the session description protocol
        u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
        e=mjh@isi.edu (Mark Handley)
        c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
        a=recvonly
        m=audio 49170 RTP/AVP 0
```

```
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

A configuration parameter **media-type** is used to specify the media group on which to operate. It contains all of the descriptors including the m-line up to the next m-line. This parameter is a string field and must match the media-type exactly as it appears within the SDP. The special media-type media can be used to refer to all media types. This is particularly useful when performing an add operation, when the user wants to add a media section between the first and second medias, but does not know what media type they are. Otherwise, during an add operation, the media section would be added before the specified media-type (if no index parameter was provided).

The types of descriptors used at the sdp-media-rule level are m, i, c, b, k, and a, the descriptors specific to the media description.

This level of granularity affords the user a very simple way to making subtle changes to the media portion of the SDP. For instance, it is very common to have to change the name of an audio format (for example G729 converted to g729b), or to add attributes specific to a certain media-type.

The index operator is supported for the media-type parameter (for example, media-type audio[1]). Like header rules, if no index is supplied, this means operate on all media-types that match the given name. For specifying specific media-types, the non-discrete indices are also supported (for example, ^ - last). Adding a media-type, without any index supplied indicates that the media should be added at the beginning. The special media-type media uses the index as an absolute index to all media sections, while a specific media-type will index relative to that given media type.

For sdp-media-rules set to an action of add where the media-type is set to media, the actual media type is obtained from the new-value, or more specifically, the string after m= and before the first space.

Given the following SDP:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
m=audio 49170 RTP/AVP 0
m=audio 48324 RTP/AVP 8
m=video 51372 RTP/AVP 31
```

With the sdp-media-rule:

```
sdp-media-rule
        name                    smr
        media-type              audio[1]
        action                  manipulate
        comparison-type         case-sensitive
        match-value
        new-value               "m=audio 1234 RTP/AVP 8 16"
```

This rule operates on the 2nd audio line, changing the port and adding another codec, resulting in the SDP:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
```

```
m=audio 49170 RTP/AVP 0
m=audio 1234 RTP/AVP 8 16
m=video 51372 RTP/AVP 31
```

The following rule, however:

```
sdp-media-rule
        name                        smr
        media-type                  media[1]
        action                      add
        comparison-type             case-sensitive
        match-value
        new-value                   "m=video 1234 RTP/AVP 45"
```

adds a new video media-type at the 2nd position of all media-lines, resulting in the SDP:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
m=audio 49170 RTP/AVP 0
m=video 1234 RTP/AVP 45
m=audio 48324 RTP/AVP 8
m=video 51372 RTP/AVP 31
```

sdp-line-rule

Unlike header-rules, sdp descriptors are not added in the order in which they are configured. Instead they are added to the SDP adhering to the grammar defined by RFC 4566 (as is shown below).

```
Session description
   v=  (protocol version)
   o=  (originator and session identifier)
   s=  (session name)
   i=* (session information)
   u=* (URI of description)
   e=* (email address)
   p=* (phone number)
      c=* (connection information -- not required if included in
        all media)
   b=* (zero or more bandwidth information lines)
   One or more time descriptions ("t=" and "r=" lines; see
        below)
   z=* (time zone adjustments)
   k=* (encryption key)
   a=* (zero or more session attribute lines)
   Zero or more media descriptions (see below)

Time description
   t=  (time the session is active)
   r=* (zero or more repeat times)

Media description, if present
   m=  (media name and transport address)
   i=* (media title)
   c=* (connection information -- optional if included at
        session level)
   b=* (zero or more bandwidth information lines)
   k=* (encryption key)
   a=* (zero or more media attribute lines)
```

* after the equal sign denotes an optional descriptor.

This hierarchy is enforced meaning that if you configure a rule which adds a session name descriptor followed by a rule which adds a version descriptor, the SDP will be created with the version descriptor first, followed by the session name.

The only validation that will occur is the prevention of adding duplicate values. In much the same way that header-rules prevents the user from adding multiple To headers, the descriptor rule will not allow the user to add multiple descriptors; unless multiple descriptors are allowed, as is in the case of b, t, r and a.

There exists a parameter **type** under the sdp-line-rule object that allows the user to specify the specific line on which to perform the operation. For example: v, o, s, i, u, e, p, c, b, t, r, z, k, a, and m. Details on these types can be found in RFC 4566.

For those descriptors, of which there may exist zero or more (b, t, r, and a) entries, indexing grammar may be used to reference the specific instance of that attribute. This indexing grammar is consistent with that of header-rules for referring to multiple headers of the same type.

Given the example SDP below:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
r=604800 3600 0 90000
r=7d 1h 0 25h
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

and the following sdp-line-rule:

```
sdp-line-rule
   name                 removeRepeatInterval
   type                 r[1]
   action               delete
```

The rule removeRepeatInterval removes the second repeat interval descriptor within the SDP.

The behavior of all SDP rules follow the same behavior of all manipulation rules in that they are executed in the order in which they are configured and that each rule executes on the resultant of the previous rule.

Each descriptor follows its own grammar and rules depending on the type specified. The values of the descriptor are evaluated at runtime since the new-values themselves are evaluated at runtime. At this time no validation of the grammar for each of the types is performed. The user is responsible for properly formatting each of the descriptors according to their specifications.

For instance, the version (v) descriptor can be removed from the SDP but leaving all descriptors for that SDP, causing the SDP to become invalid. This is consistent with the way header-rules operate, in that there is no validation for the specific headers once they have been manipulated through HMR.

Regular Expression Interpolation

An interpolated regular expression is a regular expression that is compiled and evaluated at runtime. Today all regular expressions are compiled at configuration time in order to improve performance. There are cases where a regular expression is determined dynamically from data within a SIP message. In these circumstances the regular expression is unknown until the time of execution.

In order to have a regular expression be interpolated at runtime, it must be contained within a set of {}. An interpolated expression can have any number of regular expressions and strings appended together. Any characters to the left or right of the curly braces will be appended to the value within the curly braces. The curly braces are effectively two operators treated as one (interpolate the value contained within and then concatenate the values to the left and right of the curly braces). If the comparison-type is set to pattern-rule and the match-value contains a value that matches the grammar below, then it will be treated as an interpolated expression.

([^\\]|^)\{\$[^0-9]+[^}]*\}

The example below demonstrates using a user defined variable within a regular expression of another rule at runtime.

```
element-rule

        name                            someRule
        type                            header-value
        action                          replace
        comparison-type                 pattern-rule
        match-value                     ^sip:{$rule1.$0}@(.+)$
        new-value                       sip:bob@company.com
```

If the value of $rule1.$0 evaluates to alice then it will successfully match against the string sip:alice@comcast.net. An interpolated expression can be as simple as “{$rule1.$0}” or as complex as ^sip:{rule1.$0}@{$rule2[1].$2}$. It is not possible to interpolate a normal regular expression since the grammar will not allow the user to enter such an expression. Only variables can be contained with the curly braces.

The resultant of interpolated expressions can be stored in user defined variables. Given the same example from above, if the rule someRule was referenced by another rule, the value of sip:alice@comcast.net would be stored within that rule.

Interpolation only makes a single pass at interpolation, but does so every time the Rule executes. In other words, if the Rule is applied to the Route header, it will interpolate again for each Route header instance. What this means is that the value within the curly braces will only be evaluated once. For instance, if the value {$someRule.$1} evaluates to {$foobar.$2} the Oracle USM (USM) will treat $foobar.$2 as a literal string which it will compile as a regular expression. The USM will not recursively attempt to evaluate $foobar.$2, even if it was a valid user defined variable.

Interpolated regular expressions will evaluate to TRUE if an only if both the regular expression itself can be compiled and it successfully matches against the compared string.

Regular Expressions as Boolean Expressions

Regular expressions can be used as boolean expressions today if they are the only value being compared against a string, as is shown in the case below.

```
mime-rule
        name                            someMimeRule
        content-type                    application/text
        action                          replace
        comparison-type                 pattern-rule
        match-value                     ^every good boy .*
        new-value                       every good girl does fine
However, regular expressions can not be used in conjunction with other boolean
expressions to form more complex boolean expressions, as is shown below.

mime-rule
     name                       someMimeRule
     content-type               application/text
     action                     replace
     comparison-type            boolean
     match-value                $someRule & ^every good boy .*
     new-value                  every good girl does fine
```

There are many cases where the user has the need to compare some value as a regular expression in conjunction with another stored value. It is possible to perform this behavior today, however it requires an extra step in first storing the value with the regular expression, followed by another Manipulation Rule which compares the two boolean expressions together (e.g. $someRule & $someMimeRule).

In order to simplify the configuration of some sip-manipulations and to make them more efficient this functionality is being added.

Unfortunately, it is not possible to just use the example as is shown above. The problem is there are many characters that are commonly used in regular expressions that would confuse the HMR expression parser (such as $, and +). Therefore delimiting characters need to be used to separate the regular expression from the other parts of the expression.

To treat a regular expression as a boolean expression, it needs to be enclosed within the value $REGEX(<expression>,<compare_string>=$ORIGINAL); where <expression> is the regular expression to be evaluated. <compare_string> is the string to compare against the regular expression. This second argument to the function is defaulted to $ORIGINAL which is the value of the of the specific Manipulation Rule object. It can be overridden to be any other value the user desires.

The proper configuration for the example above to use regular expressions as boolean expressions is

```
mime-rule
        name             someMimeRule
        content-type     application/text
        action           replace
        comparison-type  boolean
        match-value      $someRule & $REGEX(“^every good boy .*”)
        new-value        every good girl does fine
```

It is also possible to use expressions as arguments to the $REGEX function. These expressions will in turn be evaluated prior to executing the $REGEX function. A more complex example is illustrated below.

```
header-rule
        name              checkPAU
        header-name       request-uri
        action            reject
        comparison-type   boolean
        match-value       (!$REGEX($rule1[0],$FROM_USER))&
```

```
                    (!$REGEX($rule2[0],$PAI_USER))
        msg-type    request
        new-value   403:Forbidden
        methods     INVITE,SUBSCRIBE,MESSAGE,PUBLISH,
                    OPTIONS, REFER
```

It should be noted that when using $REGEX() in a boolean expression, the result of that expression is not stored in the user variable. The comparison-type must be set to pattern-rule in order to store the result of a regular expression.

The arguments to the $REGEX() function are interpolated by default. This is the case since the arguments themselves must be evaluated at runtime. The following example is also valid.

```
mime-rule
        name                someMimeRule
        content-type        application/text
        action              replace
        comparison-type     boolean
        match-value         $someRule & $REGEX(“^every good
                            {$rule1[0].$0} .*”)
```

Moving Manipulation Rules

Users can move rules within any manipulation-rule container. Any manipulation rule which contains sub-rules will now offer the ACLI command **move** <from index> <to index>. For example, given the order and list of rules below:

1. rule1

2. rule2

3. rule3

4. rule4

Moving rule3 to position 1 can be achieved by executing **move 3 1**. The resulting order will then be: rule3, rule1, rule2, rule4. A move operation causes a shift (or insert before) for all other rules. If a rule from the top or middle moves to the bottom, all rules above the bottom are shifted up to the position of the rule that was moved. If a rule from the bottom or middle moves to the top, all rules below are shifted down up to the position of the rule that was moved. Positions start from 1.

A valid from-index and to-index are required to be supplied as arguments to the move action. If a user enters a range that is out of bounds for either the from-index or to-index, the ACLI will inform the user that the command failed to execute and for what reason.

With respect to the issue of creating an invalid sip-manipulation by incorrectly ordering the manipulation rules, this issue is handled by the Oracle USM validating the rules at configuration time and treating them as invalid prior to runtime. This may or may not affect the outcome of the sip-manipulation as a configured rule may not perform any operation if it refers to a rule that has yet to be executed. It is now the user’s responsibility to reorder the remaining rules in order to make the sip-manipulation valid once again.

It is important to note that rules of a different type at the same level are all part of the same list. To clarify; header-rules, mime-rules, mime-isup-rules and mime-sdp-rules all share the same configuration level under sip-manipulation. When selecting a move from-index and to-index for a header-rule, one must take into consideration the location of all other rules at the same level, since the move is relative to all rules at that level, and not relative to the particular rule you have selected (for example, the header-rule).

Since the list of rules at any one level can be lengthy, the **move** command can be issued one argument at a time, providing the user with the ability to select indices. For instance, typing **move** without any arguments will present the user with the list of all the rules at that level. After selecting an appropriate index, the user is then prompted with a to-index location based on the same list provided.

For Example:

```
ORACLE(sip-mime-sdp-rules)# move
select a rule to move

1: msr sdp-type=any; action=none; match-value=; msg-type=any

2: addFoo header-name=Foo; action=none; match-value=; msg-type=any

3: addBar header-name=Bar; action=none; match-value=; msg-type=any

selection: 2
destination: 1
Rule moved from position 2 to position 1
ACMEPACKET(sip-mime-sdp-rules)#
```

Rule Nesting and Management

There will be cases where the user wants to take a stored value from the SDP and place it in a SIP header, and vice-versa. All header-rules, element-rules, mime-rules, mime-isup-rules, isup-param-rules, mime-header-rules and mime-sdp-rules are inherited from a Manipulation Rule. A Sip Manipulation is of type Manipulation which contains a list of Manipulation Rules. Each Manipulation Rule can itself contain a list of Manipulation Rules. Therefore when configuring manipulation rules, they will be saved in the order which they have been configured. This is different from the way other configuration objects are configured. Essentially, the user has the option of configuring which type of object they want and when they are done, it gets added to the end of the sip-manipulation, such that order is preserved. This will mean that any Manipulation Rule at the same level can not share the same name. For example, names of header-rules can't be the same as any of the mime-sdp-rule ones or mime-isup-rule. This allows the user to reference stored values from one rule type in another at the same level.

ACLI Configuration Examples

The following eight sections provide sample SDP manipulations.

Remove SDP

```
sip-manipulation
        name                    stripSdp
        description             remove SDP from SIP message
        mime-sdp-rule
                name                    sdpStrip
                msg-type                request
                methods                 INVITE
                action                  delete
                comparison-type         case-sensitive
                match-value
                new-value
```

Remove Video from SDP

```
sip-manipulation
        name                        stripVideo
        description                 strip video codecs from SIP
                                    message
        mime-sdp-rule
                name                    stripVideo
                msg-type                request
                methods                 INVITE
                action                  manipulate
                comparison-type         case-sensitive
                match-value
                new-value
                sdp-media-rule
                        name                removeVideo
                        media-type          video
                        action              delete
                        comparison-type     case-sensitive
match-value
                        new-value
```

Add SDP

```
sip-manipulation
        name                        addSdp
        description                 add an entire SDP if one does
                                    not exist
        mime-sdp-rule
                name                    addSdp
                msg-type                request
                methods                 INVITE
                action                  add
                comparison-type         case-sensitive
                match-value
                new-value               “v=0\r\no=mhandley
2890844526 2890842807 IN IP4 “+$LOCAL_IP+”\r\ns=SDP Seminar\r\ni=A
Seminar on the session description protocol\r\u=http:
//www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps\r\ne=mjh@isi.edu
(Mark Handley)\r\nc=IN IP4 “+$LOCAL_IP+”\r\nt=2873397496
2873404696\r\na=recvonly\r\nm=audio 49170 RTP/AVP 0\r\n”
```

Manipulate Contacts

This rule changes the contact in the SDP to the value contained in the Contact header.

```
sip-manipulation
        name                            changeSdpContact
        description                     changes the contact in the SDP to the
value of the contact header
        header-rule
                name                            storeContact
                header-name                     Contact
                action                          store
                comparison-type                 pattern-rule
                msg-type                        request
                methods                         INVITE
                match-value
                new-value
```

```
                element-rule
                        name                            storeHost
                        parameter-name
                        type                            uri-host
                        action                          store
                        match-val-type                  ip
                        comparison-type                 pattern-rule
                        match-value
                        new-value
        mime-sdp-rule
                name                            changeConnection
                msg-type                        request
                methods                         INVITE
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                new-value
                sdp-session-rule
                        name                    changeCLine
                        action                  manipulate
                        comparison-type         case-sensitive
                        match-value
                        new-value
                        sdp-line-rule
                           name                 updateConnection
                           type                 c
                           action               replace
                           comparison-type      case-sensitive
                           match-value          $storeContact.$storeHost
                           new-value            $storeContact.$storeHost.$0
```

Remove a Codec

This rule changes the contact in the SDP to the value contained in the Contact header.

```
sip-manipulation
        name                            removeCodec
        description                     remove G711 codec if it exists
        mime-sdp-rule
                name                            removeCodec
                msg-type                        request
                methods                         INVITE
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                new-value
                sdp-media-rule
                        name                        removeG711
                        media-type                  audio
                        action                      manipulate
                        comparison-type             case-sensitive
                        match-value
                        new-value
                        sdp-line-rule
                                name                    remove711
                                type                    m
                                action                  replace
                                comparison-type         pattern-rule
                                match-value             ^(audio [0-9]
                                                        {1,5} RTP.*)( [07]
                                                        \b)(.*)$
```

```
                        new-value               $1+$3
                    sdp-line-rule
                        name                    stripAttr
                        type                    a
                        action                  delete
                        comparison-type         pattern-rule
                        match-value             ^(rtpmap|fmtp):
                                                [07]\b$
                        new-value
```

Change Codec

```
sip-manipulation
        name                            convertCodec
        description                     changeG711toG729
        mime-sdp-rule
                name                            changeCodec
                msg-type                        request
                methods                         INVITE
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                new-value
                sdp-media-rule
                        name                      change711to729
                        media-type                audio
                        action                    manipulate
                        comparison-type           case-sensitive
                        match-value
                        new-value
                        sdp-line-rule
                                name                  change711
                                type                  m
                                action                replace
                                comparison-type       pattern-rule
                                match-value           ^(audio [0-9]{4,5}
                                                      RTP/AVP.*)( 0)(.*)$
new-value            $1+" 18"+$3
                        sdp-line-rule
                                name                  stripAttr
                                type                  a
                                action                delete
                                comparison-type       pattern-rule
                                match-value           ^rtpmap:0 PCMU/
                                                      .+$
                                new-value
                        sdp-line-rule
                                name                  addAttr
                                type                  a
                                action                add
                                comparison-type       boolean
                                match-value           $change711to729.
                                                      $stripAttr
                                new-value             rtpmap:18 G729/8000
```

Remove Last Codec and Change Port

```
sip-manipulation
        name                            removeLastCodec
        description                     remove the last codec
```

```
        mime-sdp-rule
                name                            removeLastCodec
                msg-type                        request
                methods                         INVITE
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                new-value
                sdp-media-rule
                        name                        removeLast
                        media-type                  audio
                        action                      manipulate
                        comparison-type             case-sensitive
                        match-value
                        new-value
                        sdp-line-rule
                                name                    isLastCodec
                                type                    m
                                action                  store
                                comparison-type         pattern-rule
                                match-value             ^(audio )([0-9]{4,
                                                        5})( RTP/AVP
                                                        [0-9]{1-3})$
new-value
                        sdp-line-rule
                                name                    changePort
                                type                    m
                                action                  replace
                                comparison-type         boolean
                                match-value             $removeLastCodec.
$removeLast.$isLastCodec
                                new-value               $removeLastCodec.
$removeLast.$isLastCodec.$1+0+$removeLastCodec.$removeLast.
$isLastCodec.$3
```

Remove Codec with Dynamic Payload

```
sip-manipulation
        name                            removeAMR
        description                     remove the AMR and AMR-WB dynamic codecs
        mime-sdp-rule
                name                            sdpAMR
                msg-type                        request
                methods                         INVITE
                action                          manipulate
                comparison-type                 case-sensitive
                match-value
                new-value
                sdp-media-rule
name                      mediaAMR
                        media-type                  audio
                        action                      manipulate
                        comparison-type             case-sensitive
                        match-value
                        new-value
                        sdp-line-rule
                                name                isAMR
                                type                a
                                action              delete
                                comparison-type     pattern-rule
                                match-value         ^rtpmap:([0-9]
```

```
                                        {2,3}) AMR
                    new-value
sdp-media-rule
        name                            mediaIsAMR
        media-type                      audio
        action                          manipulate
        comparison-type                 boolean
        match-value                     $sdpAMR.$media
                                        AMR.$isAMR
        new-value
        sdp-line-rule
                name                    delFmtpAMR
                type                    a
                action                  delete
                comparison-type         pattern-rule
                match-value             ^fmtp:{$sdpAMR.
                                        $mediaAMR.
                                        $isAMR.$1}\b
                new-value
        sdp-line-rule
               name                  delAMRcodec
               type                  m
               action                find-replace-all
                comparison-type      pattern-rule
                match-value          ^audio [0-9]+
                                     RTP.*( {$sdpAMR.
                                      $mediaAMR.$isAMR.
```

Dialog Transparency

This section explains how to configure dialog transparency, which prevents the Oracle USM from generating a unique Call-ID and modifying dialog tags.

Overview

With dialog transparency enabled, the Oracle USM is prevented from generating a unique Call-ID and from modifying the dialog tags; the Oracle USM passes what it receives. Therefore, when a call made on one Oracle USM is transferred to another UA and crosses a second Oracle USM, the second Oracle USM does not note the context of the original dialog, and the original call identifiers are preserved end to end. The signalling presented to each endpoint remains in the appropriate context regardless of how many times a call crosses through a Oracle USM or how many Oracle USMs a call crosses.

Without dialog transparency enabled, the Oracle USM's SIP B2BUA rewrites the Call-ID header and inserted dialog cookies into the From and To tags of all messages it processes. These dialog cookies are in the following format: SDxxxxxNN-. Using these cookies, the Oracle USM can recognize the direction of a dialog. However, this behavior makes call transfers problematic because one Oracle USMs' Call-ID might not be properly decoded by another Oracle USM. The result is asymmetric header manipulation and failed call transfers.

Dialog Transparency Configuration

You set one parameter in your SIP configuration to enable dialog transparency.

- For new configurations, this feature defaults to enabled
  To enable SIP dialog transparency:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ```

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **dialog-transparency**—Enter the state of SIP dialog transparency you require for your Oracle USM. The default value is **enabled**. The valid values are:

   - enabled | disabled

Route Header Removal

This section explains how to enable the Oracle USM to disregard and strip all SIP Route headers. You set an option in a SIP interface configuration to strip all Route headers for SIP requests coming from this interface.

When the Oracle USM with this option configured receives an INVITE from an interface, it removes the route headers. However, although it removes the headers, the Oracle USM maintains backward compatibility with RFC 2543 nodes. To do so, it normalizes the request to an RFC 3261 loose routing form before it removes the headers.

Route Header Removal Configuration

The following information explains how to remove SIP route headers.

To configure SIP route header removal:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **options strip-route-headers** and press Enter. This completes the configuration of SIP route header removal.

   ```
   ORACLE(sip-interface)# options strip-route-headers
   ```

SIP Via Transparency

This section explains the inbound Via header transparency feature, which enables the Oracle USM to insert its Via header on top of the top-most Via header received from user equipment (UE). It then forwards it on to the IP Multimedia Subsystem (IMS) core with the original Via header now located as the bottom-most Via header.

The Oracle USM still replaces the Contact and other header addresses with its own, and does not pass on the core's Via headers in outbound requests.

This feature is targeted for the Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN) with SIP hosted NAT traversal support. It works with SIP NAT bridged, local-policy routed, and non-SIP NAT configurations, regardless of registration handling.

Some equipment acts as Proxy-CSCF (P-CSCF) and Serving-CSCF (S-CSCF) nodes, with the Oracle USM is located between the equipment and user endpoints. The equipment needs to see the each user endpoint's original Via header in order to perform some implicit authentication, admission, and control functions in a TISPAN-compliant model.

You enable Via header transparency on the access SIP interface. Received Via headers are saved for inclusion in requests going out another interface or session agent that does not have the parameter set, in other words, the core side. For any received SIP message where the inbound previous hop interface was enabled for Via header transparency, theOracle USM adds its own Via header as it forwards it, and it also copies the received top-most Via as the new bottom-most Via, if the outbound next hop interface/session agent is not enabled for Via header transparency. The Oracle USM also adds a received= parameter to the copied Via header, per the SIP RFC 3261.

Any message received from an interface without Via header transparency enabled, does not have the received Via header copied over to any other direction.

For HNT, where the original top-most (and only) Via header from a UE is a private/false address, the SD should still copy that false address into the core-side, and the received= parameter will contain the real Layer-3 addressing.

SIP Via Transparency Configuration

You can configure SIP Via header transparency for the access SIP interface using the ACLI.

To configure SIP Via header transparency for an access interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. You can either add support to a new SIP interface configuration or to an existing SIP interface configuration:

For a new SIP interface configuration, you can add the option by typing options, a Space, and then via-header-transparency.

```
ORACLE(sip-interface)# options via-header-transparency
```

For an existing SIP interface configuration without options configured, select the SIP interface, type options followed by a Space, and then via-header-transparency.

```
ORACLE(sip-interface)# select
ORACLE(sip-interface)# options via-header-transparency
```

For an existing SIP interface configuration with options configured, select the SIP interface, type options followed by a Space, the plus sign (+), and the via-header-transparency option.

```
ORACLE(sip-interface)# select
ORACLE(sip-interface)# options +via-header-transparency
```

5. Save your work using the ACLI **save** or **done** command.

Symmetric Latching

Symmetric latching, or forced HNT, ensures that symmetric RTP/RTCP is used for a SIP endpoint. Symmetric RTP/RTCP means that the IP address and port pair used by an outbound RTP/RTCP flow is reused for the inbound flow. The IP address and port are learned when the initial RTP/RTCP flow is received by the Oracle USM. The flow's source address and port are latched onto and used as the destination for the RTP/RTCP sourced by the other side of the call. The IP address and port in the c line and m line respectively in the SDP message are ignored.

If your network is configured with nested realms in order to separate signalling from media, make sure that the symmetric latching feature is enabled on the signaling realm.

> **Note:**
>
> This description is applicable to RTCP only when you also enable the HNT RTCP option in the media-manager configuration. Do not enable symmetric latching on core-facing interfaces.

Symmetric Latching Configuration

To configure symmetric latching:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **symmetric-latching** — identifies whether and how to enable symmetric latching on the SBC. This completes the configuration of forced Hosted NAT Traversal (HNT). The default value for this parameter is **disabled**. The valid values are:
   - **disabled** —
   - **enabled** —
   - **pre-emptive** —

   ```
   ORACLE(realm-config)# symmetric-latching pre-emptive
   ```
4. Type **done** to save your configuration.

Enabling RTCP Latching

To enable RTCP symmetric latching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```
2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```
3. Type **media-manager** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```
4. Select the media manager configuration so that you can enable HNT RTCP.

   ```
   ORACLE(media-manager-config)# select
   ```
5. **hnt-rtcp —** Enable support of RTCP when the SBC performs HNT. The default value is **disabled**. The valid values are:
   - enabled | disabled

   ```
   ORACLE(media-manager-config)# hnt-rtcp enabled
   ```
6. Save your work using either the ACLI **save** or **done** command.

SIP Number Normalization

This section explains the SIP number normalization feature that applies to the SIP To URI. (Currently the Oracle USM supports number normalization on From and To addresses for both inbound and outbound call legs.) Number normalization includes add, delete, and replace string functions that result in consistent number formats.

Number normalization is supported for the following call types:

- SIP to SIP
- H.323 to SIP

Number normalization applies to the SIP To URI. It occurs on ingress traffic, prior to the generation of accounting records or local policy lookups. RADIUS CDR attributes are populated with the normalized numbers. Local policy matching is based on the normalized numbers.

Terminology

The following lists explains the terminology used later.

- X is any digit having the value 0 through 9
- N is any digit having the value 2 through 9
- 0/1 is a digit having the value of either 0 or 1
- NXX is a form of Numbering Plan Area (NPA).
- CC is a 1, 2, or 3 digit country code used in international dialing
- NN is a national number that can be a four to fourteen digit national number used in international dialing, where the combination of CC+NN is a 7 to 15 digit number.
- + symbol in E.164 indicates that an international prefix is required
- E.164 numbers are globally unique, language independent identifiers for resources on Public Telecommunication Networks that can support many different services and protocols.
- N11 number is any of the three-digit dialing codes in the form N11 used to connect users to special services, where N is a digit between 2 and 9

Calls from IP Endpoints

The Oracle USM uses the following number normalization rules:

- North American Numbering Plan (NANP) calls: where a number with the format 1NPANXXXXXX is received, the Oracle USM adds a plus sign (+) as a prefix to the NANP number. The Oracle USM also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:+1NPANXXXXXX@ipaddr;user=phone
  ```

- International NWZ1 calls: Oracle USM receives an international call with the format 011CCNN. The Oracle USM deletes the 011 prefix and adds a plus sign (+) as a prefix to CC+NN; and also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:+CCNN@ipaddr;user=phone
  ```

- Private number calls: when a private number with the format nxxxx (where n=2 through 9) is received, no number normalization is applied by the Oracle USM.
- Calls to numbers such as N11, 0-, 0+, 00-, and 01+: the Oracle USM adds ;phone-context=+1 after the number and also adds the string ;user=phone after the host IP address in the SIP URI. For example:

  ```
  sip:N11;phone-context=+1@ipaddr;user=phone
  sip:01CCNN;phone-context=+1@ipaddr;user=phone
  ```

- Calls with numbers that are already normalized are not modified by the Oracle USM.

Calls from IP Peer Network

For calls received from external peer networks, the Oracle USM uses the following number normalization rules:

- Global numbers such as NANP and international E.164 numbers should have already been normalized. If not, the Oracle USM applies the same number normalization rules listed in the prior section.
- Calls to numbers such as N11, 0-, 0+, 00-, and 01+: the Oracle USM adds ;phone-context=+1 after the number and also adds the string ;user=phone (if absent) after the host IP address in the SIP URI.

SIP Number Normalization Configuration

You can configure SIP number normalization for the realm and session agent using the ACLI.

Realm

To configure SIP number normalization for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. You can either add SIP number normalization support to a new session agent configuration or to an existing session agent configuration:

   - For a new realm configuration, add the option by typing **options**, a Space, and then **number-normalization**.

     ```
     ORACLE(realm-config)# options number-normalization
     ```

   - For an existing realm configuration without any options already configured, select the realm, type **options** followed by a Space, and then **number-normalization**.

     ```
     ORACLE(realm-config)# select
     ORACLE(realm-config)# options number-normalization
     ```

   - For an existing realm configuration with other options, select the realm, type **options** followed by a Space, the plus sign (+), and the **number-normalization** option.

     ```
     ORACLE(realm-config)# select
     ORACLE(realm-config)# options +number-normalization
     ```

5. Save your work using the ACLI **save** or **done** command.

Session Agent

To configure SIP number normalization for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. You can either add SIP number normalization support to a new session agent configuration or to an existing session agent configuration:

   - For a new a session agent configuration, add the option by typing **options**, a Space, and then **number-normalization**.

     ```
     ORACLE(session-agent)# options number-normalization
     ```

   - For an existing session agent configuration without any options already configured, select the session agent, type **options** followed by a Space, and then **number-normalization**.

     ```
     ORACLE(session-agent)# select
     ORACLE(session-agent)# options number-normalization
     ```

   - For an existing session agent configuration with other options, select the session agent, type **options** followed by a Space, the plus sign (+), and the **number-normalization** option.

     ```
     ORACLE(session-agent)# select
     ORACLE(session-agent)# options +number-normalization
     ```

5. Save your work using the ACLI **save** or **done** command.

SIP Port Mapping

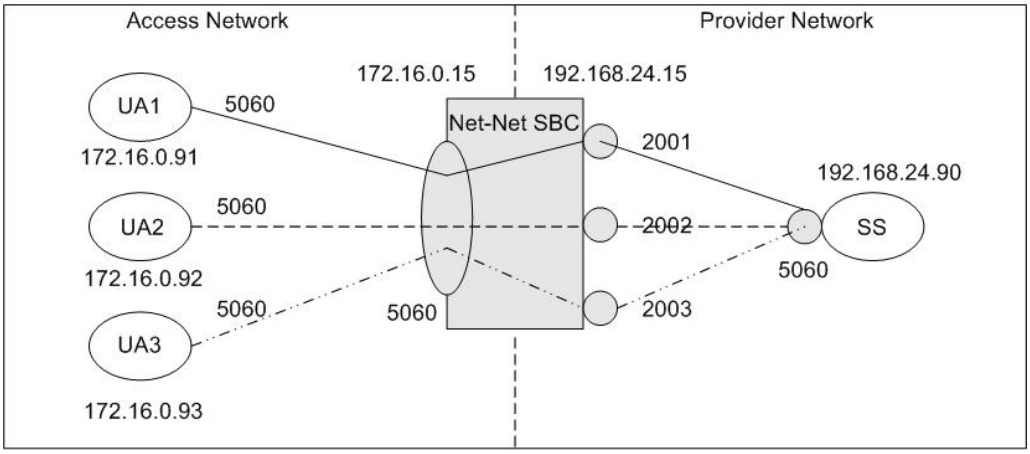
This section contains information about the SIP port mapping feature. SIP port mapping lets you allocate a unique SIP signaling transport address (IP address and UDP port) on the Oracle USM in the provider network for each registered endpoint (user agent).

About SIP Port Mapping

You might need to provide a unique signaling transport address for each registered endpoint for admission control, if required by your softswitch vendor. If you have questions about your softswitch, contact the vendor for assistance.

When a Oracle USM resides between the endpoints and the softswitch, the softswitch sees the same transport address (that of the Oracle USM) for all endpoints. By allocating a unique UDP port for each endpoint, the Oracle USM provides each of them a unique transport address.

The following example illustrates the SIP port mapping feature.



The diagram shows UA1, UA2, and UA3 are endpoints within the access network and that the SIP interface for the access network is 172.16.0.15:5060. On the provider network, the SIP interface is at 192.168.24.15, with the SIP port mapping feature enabled. The softswitch/ registrar is also located on the provider network at 192.168.24.90:5060.

The diagram shows that port 2001 on the provider network is allocated to UA1 on the access network, port 2002 is allocated to UA2, and port 2003 is allocated to UA3. Because of this allocation, all SIP signaling messages sent from the endpoints in the access network to the softswitch on the provider network travel through an allocated signaling port. For example, all signaling messages between UA1 and the softswitch use 192.168.24.15:2001 as the transport address.

How SIP Port Mapping Works

The Oracle USM (USM) allocates SIP port mapping (signaling) ports during a REGISTER request that has registration caching applied. When you define a range of signaling ports for the SIP interface, you create a pool of signaling ports that can be allocated during the REGISTER request.

The USM allocates a signaling port from the pool when it creates the registration cache entry for a Contact in a REGISTER request. It allocates a separate signaling port for each unique Contact URI from the access side. The registration cache Contact entry contains the mapping between the Contact URI in the access/endpoint realm (the UA-Contact) and the Contact URI in the registrar/softswitch realm (the SD-Contact).

The SD-Contact is the allocated signaling port. The signaling port gets returned to the pool when the Contact is removed from the registration cache. The removal can occur when the cache entry expires; or when the endpoint sends a REGISTER request to explicitly remove the Contact from the registrar. When a signaling port returns to the pool it gets placed at the end of pool list; in a least-recently-used allocation method for signaling ports

When the USM forwards the REGISTER request to the softswitch, it replaces the UA-Contact with SD-Contact. For example, if UA1 sends a REGISTER request with a Contact URI of sip:ua1@172.16.0.91:5060, it is replaced with sip:192.168.24.15:2001 when the REGISTER request is forwarded to the registrar.

The same translation occurs when UA1 sends that same URI in the Contact header of other SIP messages. SIP requests addressed to the allocated signaling transport address (SD-Contact) are translated and forwarded to the registered endpoint contact address (UA-Contact).

> **Note:**
>
> The maximum number of registered endpoints cannot exceed the number of signaling ports available. If no signaling ports are available for a new registration, the REGISTER request receives a 503 response.

The USM still processes requests received on the configured SIP port address. Requests sent into the registrar/softswitch realm that are not associated with a registered user will use the configured SIP port address.

Using SIP port mapping with SIPconnect—where unique ports are used for each registered PBX—hinders the USM from routing incoming calls to the corresponding PBX because the USM uses DN for the PBX's parent during registration, but the incoming INVITE from the softswitch contains the child DN in its Request URI. Thus the USM cannot find a matching SBC-Contact because the username of the Request URI contains the child DN, but the username of the SBC-Contact contains the parent DN.

You can enable SIPconnect support in either the realm configuration or session agent for the SIP access network by setting the **sip-connect-pbx-reg** option. With this option set and the destination realm configured for port mapping, the USM inserts a special search key in the registration table. Rather than adding the SD-Contact as the key as with regular (non-SIPconnect) registrations, the USM strips user information and instead uses the host and port information as the registration key. The USM still forwards the registration message with an intact contact username.

SIP Port Mapping Based on IP Address

Some registrars need to know that multiple contacts represent the same endpoint. The extension to this feature answers the expectation from registrars that an endpoint registering multiple AoRs will use a single core-side mapped port to show that the AoRs really represent a single endpoint.

When you enable SIP port mapping based on IP Address, the Oracle USM supports core-side UDP port mapping based on the endpoint's IP address. It ignores the username portion of the AoR or Contact.

The Oracle USM performs the port mapping allocation and lookup based on all requests using the via-key from the SIP Request. The via-key is a combination of Layer 3 and Layer 5 IP information in the message. The Oracle USM performs an additional lookup in the registration table to determine if a via-key already exists. If it does, then the Oracle USM uses the port already allocated and does not allocate a new one.

About NAT Table ACL Entries

To enable SIP signaling messages to reach the host processor, the Oracle USM adds NAT table ACL entries for each SIP interface. With UDP without SIP port mapping applied, it adds a single ACL entry for each SIP port in the SIP interface configuration. For example:

```
untrusted entries:
intf:vlan source-ip/mask:port/mask dest-ip/mask:port/mask   prot type    index
0/0:0     0.0.0.0                  172.16.1.15:5060         UDP  static  10
0/3:0     0.0.0.0                  192.168.24.15:5060       UDP  static  16
0/1:0     0.0.0.0                  192.168.50.25:5060       UDP  static  17
```

Using SIP Port Mapping

When you use SIP port mapping, one or more ACL entries are added to the NAT table to enable the range of ports defined. The NAT table does not support the specification of port ranges. However, it does support masking the port to enable ranges that fall on bit boundaries. For example, an entry for 192.168.24.15:4096/4 defines the port range of 4096 through 8191.

The algorithm for determining the set of ACLs for the port map range balances the need to represent the range as closely as possible, with the need to minimize the number of ACL entries. For example, a range of 30000 through 39999 would result in the following set of ACLs.

```
untrusted entries:
intf:vlan source-ip/mask:port/mask dest-ip/mask:port/mask   prot type    index
0/3:0     0.0.0.0                  192.168.24.15:30000/4    UDP  static  13
0/3:0     0.0.0.0                  192.168.24.15:32768/4    UDP  static  14
0/3:0     0.0.0.0                  192.168.24.15:36864/4    UDP  static  15
```

However, the first entry actually enables ports 28672 though 32767 and the last entry allows port 36864 through 40959. If SIP messages are received on ports outside the configured range (28672 through 29999 or 40000 through 40959 in this case), they are ignored.

Acme Packet recommends you use port map ranges that fall on bit boundaries to ensure the fewest possible ACL entries are created and only the configured ports are allowed by the ACLs. For example, a range of 32768 to 49151 provides for 16,384 signaling ports in a single ACL entry (192.168.24.15:32768/2).

> **Note:**
>
> If the ACLs added for the port map range do not include the SIP port configured in the SIP interface; the normal SIP ACL entry for the SIP port is also added.

Dynamic Configuration

Dynamic configuration of SIP port mapping can cause disruption in service for existing registration cache entries; depending on the changes made to the defined port map range. If the range of mapping ports is reduced, it is possible that SIP signaling messages from the registrar/softswitch realm will no longer be sent to the host processor because of the changes in the NAT Table ACL entries.

When the range of mapping ports is changed, any signaling ports in the free signaling port pool not allocated to a registration cache entry are removed from the pool. When an allocated signaling port that is no longer part of the defined mapping port range is released, it is not returned to the pool of free steering ports.

The administrator is warned when the changed configuration is activated after the port map range of a SIP interface has been changed.

Registration Statistics

The SIP registration cache statistics include counters for free and allocated signaling ports. You can issue a show registration command to display the statistics:

```
17:36:55-190
SIP Registrations          -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
User Entries           4       4       0          7       4       4
Local Contacts         4       4       0          7       4       4
Free Map Ports     12284   12284       0      12291   12288   12288
Used Map Ports         4       4       0          7       4       4
Forwards               -       -       1         22       4
Refreshes              -       -       3         43       3
Rejects                -       -       0          0       0
Timeouts               -       -       0          1       1
Fwd Postponed          -       -       0          0       0
Fwd Rejected           -       -       0          0       0
Refr Extension         0       0       0          0       0       0
Refresh Extended       -       -       0          0       0
```

The labels for the first two items reflect the restructured registration cache:

- User Entries: counts the number of unique SIP addresses of record in the cache. Each unique address of record represents a SIP user (or subscriber). The address of record is taken from the To header in the REGISTER request. There might be one or more registered contacts for each SIP user. The contacts come from the Contact header of the REGISTER request.
- Local Contacts: counts the number of contact entries in the cache. Because the same user can register from multiple endpoints (user agents); the number of Local Contacts might be higher than the number of User Entries.
- Free Map Ports: counts the number of ports available in the free signaling port pool.
- Used Map Ports: counts the number of signaling ports allocated for registration cache entries. The value of Used Map Ports will equal the number of Local Contacts when the port mapping feature is used for all registrar/softswitch realms in the Oracle USM.

SIP Port Mapping Configuration

You configure the SIP port mapping feature on a per-realm basis using the SIP interface configuration. Configure the port map range on the SIP interface for the realm where the registrar/softswitch resides. Port mapping is only applied when the access/ingress realm has registration caching and/or HNT enabled.

The range of SIP mapping ports must not overlap the following:

- Configured SIP port, which might be used for signaling messages not associated with a registered endpoint.
- Port range defined for steering pool configuration using the same IP address as the SIP interface. If overlap occurs, the NAT table entry for the steering port used in a call prevents SIP messages from reaching the host processor.
  To configure SIP port mapping:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **port-map-start**—Set the starting port for the range of SIP ports available for SIP port mapping. The valid range is 1025 through 65535. The default values is **0** and when this value is set, SIP port mapping is disabled. The valid range is:

   - Minimum: 0, 1025
   - Maximum: 65535

   ```
   ORACLE(sip-interface)# port-map-start 32768
   ```

5. **port-map-end**—Set the ending port for the range of SIP ports available for SIP port mapping. The valid range is 1025 through 65535. If you set the value to the default **0**, SIP port mapping is disabled. The valid range is:

   - Minimum—0, 1025
   - Maximum—65535

   > **Note:**
   >
   > If not set to zero (0), the ending port must be greater than the starting port.

   ```
   ORACLE(sip-interface)# port-map-end 40959
   ```

6. **options**—If you want to use SIP port mapping based on IP address, set the options parameter by typing **options**, a Space, the option name **reg-via-key** with a plus sign in front of it, type the equal sign and the word **all**. Then press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-key=all
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

7. Save your work using the ACLI **done** command.

   The following example shows SIP port mapping configured for a SIP interface:

```
sip-interface
        state                           enabled
        realm-id                        backbone
        sip-port
                address                         192.168.24.15
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 all
        sip-port
                address                         192.168.24.15
                port                            5060
                transport-protocol              TCP
                allow-anonymous                 all
        carriers
        proxy-mode
        redirect-action
        contact-mode                    none
        nat-traversal                   none
        nat-interval                    30
```

```
registration-caching          enabled
min-reg-expire                120
registration-interval         3600
route-to-registrar            enabled
teluri-scheme                 disabled
uri-fqdn-domain
trust-mode                    agents-only
max-nat-interval              3600
nat-int-increment             10
nat-test-increment            30
sip-dynamic-hnt               disabled
stop-recurse                  401,407
port-map-start                32768
port-map-end                  40959
last-modified-date            2005-09-23 14:32:15
```

SIP Port Mapping for TCP and TLS

In releases prior to S-C6.2.0, the Oracle USM (USM) supports SIP port mapping for UDP and now you can enable this feature for SIP sessions using TCP and TLS. Port mapping enables the USM to allocate a unique port number for each endpoint registering through it by giving it a transport address (or hostport) in the registered Contact.

When you enable this feature for TCP and TLS, the USM designates a port from a configured range for each endpoint that registers with SIP servers in the SIP interface's realm. You establish that range of ports using the **port-map-start** and **port-map-end** parameters. Unlike its behavior with UDP port mapping—where the USM sends requests on the SIP interface from the allocated port mapping, the USM sends all requests over an existing connection to the target next hop for TCP/TLS port mapping. If a connection does not exist, the system creates one. So for TCP/TLS port mapping, only the Contact header contains the transport address of the mapping port (i.e., the transport address of the configured SIP port). And the system refuses TCP and TLS connections on the allocated mapping port.

With TCP/TLS port mapping enabled, the USM sends the Path header with the transport address in Register requests, unless you specify that it should not do so. Standards-conformant SIP servers (that support RFC 3327) might attempt to send requests to the allocated mapping port if the Path header is absent.

> **Note:**
>
> ACL entries in the NAT table that permit TCP/TLS signaling for a SIP port configuration with TCP/TLS port mapping are the same as they would be for a TCP/TLS SIP port without port mapping enabled. Additional ACL entries that need to be set up for UDP port mapping are not required for TCP/TLS port mapping.

RTN 1684

SIP Port Mapping Configuration for TCP TLS

You enable TCP/TLS port mapping in a per-realm basis using the SIP interface configuration; setting the **tcp-port-mapping** value in the **options** parameter enables the feature. Enabling this parameter turns on the port mapping feature for UDP as well.

By default, the Oracle USM includes the Path header in Register requests it sends from that SIP interface. If you do not this header to be included, however, you can set the value as **tcp-port-mapping=nopath**.

To enable TCP/TLS port mapping for a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name **tcp-port-mapping** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +tcp-port-mapping
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

SIP Configurable Route Recursion

When the Oracle USM routes SIP requests from a UAC to a UAS, it might determine that there are multiple routes to try based on a matching local policy. The Oracle USM recurses through the list of routes in a specific order according to your configuration and the quality of the match. There are other scenarios when a UAS replies with a 3xx Redirect response to the Oracle USM, the 3xx response can include multiple Contacts to which the request should be forwarded in a specific order. In both cases, the Oracle USM needs to recurse through a list of targets.

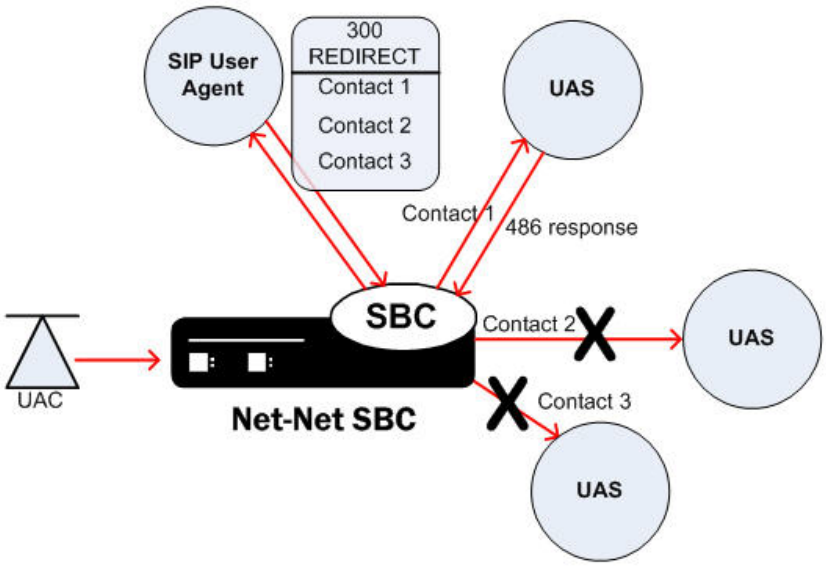
When the Oracle USM receives a non-successful (or non-6xx response) final response from the UAS, and there are multiple targets for the original request, the Oracle USM will forward the request to the next target and wait for a response. While the process of forwarding the request to multiple targets as explained in the previous paragraph is called serial forking, and the process of forwarding the request to contacts received in redirect responses is called recursion, the term recursion is used for both processes in this notice.

Use the SIP Route Recursion feature when you want the Oracle USM to forward a response to the UAC and stop recursing through the target list immediately after receiving the 3xx, 4xx, or 5xx response code that you configure. When this feature is disabled, the Oracle USM only stops recursing when it receives a message with a 401 or 407 response code. Using this feature, you can configure a specific message or range of messages to stop recursing on when received. The Oracle USM retains its default behavior to stop recursing on a 401 or 407 response code when SIP Route Recursion is configured on a SIP interface. The Oracle USM will always stop recursing when it receives a global failure (6xx); this behavior is not configurable.

You can disable response recursion for either a SIP interface or for a SIP session agent, providing you with flexibility for various network architectures. For instance, a PSTN gateway might be the only hop to reach a given endpoint, whereas several session agents might need to be contacted if multiple devices map to a contacted address of record.

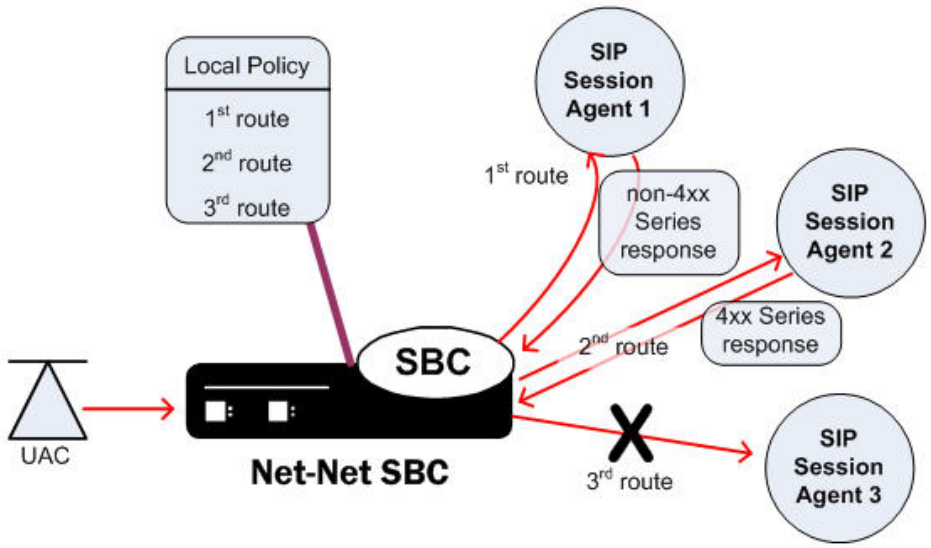
Example 1

A more detailed example is when a softswitch might return a list of contacts for multiple PSTN gateways in a Redirect message. If the PSTN target number contacted on redirection is busy, a 486 response will be sent to the Oracle USM. Since the single target is located in the PSTN, a subsequent request through a different gateway will yield another 486 response. The Oracle USM should be configured to return the 486 response to the UAC immediately. No other SIP requests should be sent to applicable targets/contacts that were enumerated in the redirect list. See the following example:



Example 2

The Oracle USM might determine from a local policy lookup that several routes are applicable for forwarding a SIP message. The Oracle USM will try each route in turn, but the SIP response recursion disable feature can be implemented to stop the route recursion when a configured responses message is received by the Oracle USM. See the following example:



There are a few conditions on the parameter used to configure response recursion:

- SIP Route Recursion is configurable for either the SIP interface or session agent.
- 401 and 407 are preconfigured for all configured SIP interfaces. They are not configured for session agents.
- The format is a comma-separated list of response codes or response code ranges: 404, 484-486.
- Only response codes that fall within the 3xx, 4xx, and 5xx range may be specified.

SIP Route Recursion Configuration

You enable SIP route recursion either in the session agent or the SIP interface configuration.

Configuring a Session Agent for SIP Route Recursion

To configure SIP Route recursion for an existing session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Select the session agent where you want this feature.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: asd          realm=      ip=1.0.0.0
   ```

```
2: SIPSA        realm=      ip=10.10.102.1
selection:2
ORACLE(session-agent)#
```

5. **stop-recurse**—Enter list of returned response codes that this session agent will watch for in order to stop recursion on the target's or contact's messages. This can be a comma-separated list or response code ranges.

   ```
   ORACLE(session-agent)# stop-recurse 404,484-486
   ```

6. Save and activate your changes.

Configuring a SIP Interface for SIP Route Recursion

To configure SIP route recursion for an existing SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Select the SIP interface to which you want to apply this feature.

   ```
   ORACLE(sip-interface)# select
   <realm-id>:
   1: Acme_Realm
   selection:1
   ORACLE(sip-interface)#
   ```

5. **stop-recurse**—Enter a list of returned response codes that this SIP interface will watch for in order to stop recursion on the target's or contact's messages. This list can be a comma-separated list of response codes or response code ranges.

   ```
   ORACLE(sip-interface)# stop-recurse 404,484-486
   ```

6. Save and activate your changes.

SIP Event Package Interoperability

Service providers often deploy a Oracle USM on the border of an access network, where it sits between the SIP endpoints (user agents) and the service provider's application server. The application server and the user agents sometimes use various SIP event packages to exchange and maintain state information. The SUBSCRIBE and NOTIFY methods are used to establish subscriptions to the event packages and to report state changes to the subscribing entity.

The SIP global contact option addresses interoperability in the Dialog and Presence event packages that are used in hosted PBX and IP Centrex offerings. State information is passed in the message body of a NOTIFY request; this message body is encoded in an XML format described by the Content-Type header. The Oracle USM needs to update certain fields in the body to account for dialog mapping and SIP NAT functionality between the access and service provider realms. Often the subscriptions are established using URIs learned from Contact headers in the user agent registrations or dialog establishment (INVITE/SUBSCRIBE). For

this, a Oracle USMrequires a Contact URI that is usable and routable outside of an existing dialog.

The SIP global contact option enables persistent URIs in the Contact headers inserted into outgoing SIP messages. If this option is not used, URIs placed in the Contact header of outgoing messages are only valid within the context of the dialog to which the message is associated.

RFCs associated with this feature are:

- A. B. Roach, *Session Initiation Protocol (SIP)-Specific Event Notification*, RFC 3265, June 2002
- J. Rosenberg, *A Presence Event Package for the Session Initiation Protocol (SIP)*, RFC 3856, August 2004
- J. Rosenberg, et al. *Data Format for Presence Using XML*, http://www.iptel.org/info/players/ietf/presence/outdated/draft-rosenberg-impp-pidf-00.txt, Work In Progress (expired), June 2000
- J.Rosenberg, H. Schulzrinne, R. Mahy, *An INVITE Initiated Dialog Event Package for the Session Initiation Protocol (SIP)*, draft-ietf-sipping-dialog-package-06.txt, Work In Progress, April 2005
- H. Sugano, et al., *Presence Information Data Format (PIDF)*, RFC 3863, August 2004

SIP Event Package Interoperability Configuration

This feature is applicable to the global SIP configuration.

To configure SIP event package interoperability:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Add SIP event package interoperability support to a new SIP configuration or to an existing SIP configuration:

   If you do not currently have an SIP configuration, you can add the option by typing options, a Space and then global-contact.

   ```
   ORACLE(sip-config)# options global-contact
   ```

   Select the SIP configuration so that you can add SIP event package interoperability support to it. Then, to add this option to a list of options that you have already configured, type options followed by a Space, the plus sign (+), and the global-contact option.

   ```
   ORACLE(sip-config)# select
   ORACLE(sip-config)# options +global-contact
   ```

If you type options global-contact without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your changes.

SIP Proxy Subscriptions

When the Oracle USM operates in dialog mode (i.e., as a B2BUA), it creates and maintains dialog state for subscription dialogs created with SUBSCRIBE/NOTIFY messages and for INVITE-initiated dialogs. Since there can be a very large number of subscriptions per user in a Rich Communication Services (RCS) environment (especially for presence subscriptions), Oracle USM resources can quickly become depleted.

To alleviate this consumption of resources, you can configure your **Oracle USM** to operate in proxy mode for event packages that you define using the **proxy-sub-event** parameter in the global SIP configuration. When you define event packages in this list and the operation mode for the SIP configuration is dialog or session, the **Oracle USM** processes all SUBSCRIBE and NOTIFY requests and responses for the designated event packages in transaction stateful mode.

Topology Hiding

So that it can perform topology hiding, the Oracle USM retains necessary routing information (such as the Contact or Record-Route header values) and it encodes certain data from the messages it receives in the messages it sends. To be more specific, the Oracle USM encodes the original URI hostport and the ingress realm name into a gr parameter it adds to the URI. The hostport information is replaced with the IP address and port of the SIP interface from which the message is sent (the egress interface). Without this information, subsequent in-dialog messages cannot be routed correctly because the Oracle USM does not retain dialog state (i.e., the route-set or remote-target).

For example, the URI sip:td@192.168.24.121:5060 might be encoded as sip:td@192.168.24.121:5060; gr=vjml9qtd175bhmhvhkgp0jov81popvbp000040.

The Oracle USM also performs URI encoding on the message body for Content-Type application/pdif+xml. this contains a Presence Information Data Format document (or PIDF) in PUBLISH and NOTIFY requests so subsequent SIP requests using the URIs in the PIDF document can be routed correctly.

The Oracle USM performs URI encoding in outgoing SIP messages after SIP=NAT is applied and before outbound HMR occurs. And the system decodes URIs in SIP messages after inbound HMR takes place and before SIP-NAT is applied.

In the event a URI is encoded several times (as is the case in spiral and hairpin calls), the encoded realm+hostport values are separated by a plus sign (+), as in the following:

```
sip:td@192.168.24.121:5060; gr=vjml9qtd175bhmhvhkgp+dhfhb0jov81opvbp
```

When the Oracle USM receives any of the following requests, it matches the contents of the request's Event header with the list you configure in the proxy-sub-events parameter:

- PUBLISH
- SUBSCRIBE
- NOTIFY
- REFER

This is provided the operation-mode for the SIP configuration is set to either **session** or **dialog**. If it finds a match, the Oracle USM marks the request for processing in transaction-stateful mode rather than in B2BUA mode.

> **Note:**
>
> Although PUBLISH is not a dialog-creating request, topology hiding needs to be applied to the PIDF so that subsequent NOTIFY requests containing portions of the published PIDF can be decoded properly.

When the Oracle USM forwards the request, it will have encoded all Contact and Record-Route header information using the ingress realm. The hostport value of the URIs then has egress SIP interface's IP address and port. The Via headers in the requested the Oracle USM received are not included in the outgoing request.

Then PUBLISH, SUBSCRIBE, NOTIFY, and REFER responses are compared to the request that was sent to determine if the response should receive transaction-stateful proxy treatment. The Oracle USM decodes any encoded Record-Route headers back to their original values for the outgoing response. Any Record-Route headers added downstream from the Oracle USM are encoded using the original request's egress realm (meaning the realm from which the response was received). In addition, the Contact header is encoded using the request's egress realm and ingress SIP interface and port.

Feature Interaction

When using this feature, the Oracle USM does not keep dialog or subscription state. Therefore the Per-user SUBSCRIBE Dialog Limit feature—configured in the **enforcement-profile** configuration—will not function properly when a subscription is handled in proxy mode.

SIP Proxy Subscription Configuration

This section shows you how to configure a list of SIP event packages to cause the Oracle USM to act in proxy mode.

> **Note:**
>
> The operation-mode parameter for the global SIP configuration must be set to either dialog or session in order for this feature to function as designed.

To configure a list of SIP event packages to enable SIP proxy subscriptions:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **proxy-sub-events**—Enter a list of SIP event package names that you want to enable the SIP proxy subscriptions feature. You can enter more than one value by enclosing multiple values in quotations marks, as in the following example.

   ```
   ORACLE(sip-config)# proxy-sub-events presence winfo
   ```

SIP REGISTER Forwarding After Call-ID Change

This feature addresses the case when an endpoint reboots and performs a third party registration before its old registration expires. During this reregistration, the contact header is the same as it was pre-reregistration. As a consequence of the reboot, the SIP Call-ID changes. In this situation, the Oracle USM does not forward the REGISTER to the registrar, because it believes the endpoint is already registered, based on a previous registration from the same Contact: header URI.

To remedy this problem, the Oracle USM now keeps track of the Call-ID in its registration cache. The **forward-reg-callid-change** option in the global SIP configuration element forces the Oracle USM to forward a REGISTER message to the registrar when the Call-ID header changes in a REGISTER message received from a reregistering UAC.

SIP REGISTER Forwarding Configuration

To configure SIP REGISTER forwarding after a Call-ID change:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options—**Add this feature to a new or an existing SIP configuration:

   If you do not currently have a SIP configuration, you can add the option by typing options, a Space, and then forward-reg-callid-change.

   ```
   ORACLE(sip-config)# options forward-reg-callid-change
   ```

   For an existing SIP configuration, select the SIP configuration so that you can add this feature to it. Then, to add this option to a list of options that you have already configured, type options, a Space, the plus sign (+), and the forward-reg-callid-change option.

   ```
   ORACLE(sip-config)# options +forward-reg-callid-change
   ```

   If you type options forward-reg-callid-change without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your changes.

SIP Local Response Code Mapping

The SIP local response code mapping feature has been added as an enhancement to the SIP response code mapping. The SIP response code map feature lets you establish a table that maps SIP response-received messages (entries) to response-to-send messages (entries).

SIP local response code mapping is used with the SIP responses generated by the Oracle USM towards a specific SIP session agent. This feature lets you provision the mapping of the response codes used by the Oracle USM when it generates the responses towards a session agent.

You create the SIP local response code map using the existing mapping functionality, and then assigning that map to a session agent or to a SIP interface.

> **Note:**
>
> The configured response map is not used when the Oracle USM is acting as proxy for the responses to this session agent.

SIP Local Response Code Mapping Configuration

The following instructions explain how to create the SIP response code map and then how to assign it to a specific session agent.

Creating a SIP Response Code Map

To create a SIP local response code map:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-response-map** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-response-map
   ORACLE(response-map)#
   ```

4. **name**—Enter the name of the SIP response map you want to configure. This value is required and must be unique.

   ```
   ORACLE(response-map)# name busy
   ```

5. **entries**—To configure the entries for this mapping, type **entries** and then press Enter. Typing a question mark will show you the response code entry parameters that you can configure.

   ```
   ORACLE(response-map)# entries
   ORACLE(response-map-entries)#
   ```

recv-code—Enter original SIP response code for the recv-mode parameter. The valid range is:

- Minimum—100
- Maximum—699

```
ORACLE(response-map-entries)# recv-mode 486
```

xmit-code—Enter the SIP response code into which you want the original response code to be translated. This valid range is:

- Minimum—100
- Maximum—699

```
ORACLE(response-map-entries)# xmit-mode 600
```

reason—Enter a reason for the translated code into the reason parameter. This response comment is sent with the translated code. Make your entry in quotation marks.

```
ORACLE(response-map-entries)# reason Busy Everywhere
```

The following two parameters (**method** and **register-response-expires**) enable a SIP registration response mapping feature that allows you to configure the Oracle USM to remap a SIP failure response—which it receives from another network device or that it generates locally—to a 200 OK. You might want the Oracle USM to perform this type of mapping for circumstances where non-malicious endpoints continually attempt registration, but will stop (and still not be registered) when they receive a 200 OK. This response mapping does not actually register the client with the Oracle USM, meaning that there is neither a registration cache entry or a CAM ACL for it.

For the 200 OK it generates, the Oracle USM removes any Reason or Retry-After header in the 200 OK and sets the expires time. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). You can also set this value using the register-response-expires parameter, but the value you set should never exceed the Register request's expires time.

method—Enter the name of the received SIP failure response message you want to map to a 200 OK. There is no default for this parameter, and leaving the parameter empty turns off the SIP registration response mapping feature.

register-response-expires—Enter the time you want to use for the expires time what mapping the SIP method you identified in the method parameter from Step 4. The maximum is 999999999. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). Any value you configure in this parameter (when not using the defaults) should never exceed the Register request's expires time.

6. Note the name that you gave the SIP response code map so that you can use it when you configure a session agent to support SIP response code mapping.
7. Save and activate your changes.

Assigning SIP Response Code Maps to Session Agents

To assign a SIP local response code map to a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **local-response-map**—E**nter the name of the configured SIP response map that you want to use for this session-agent** and press Enter.

```
ORACLE(session-agent)# local-response-map busy
```

5. Save and activate your configuration.

Assigning SIP Response Code Maps to SIP Interfaces

To apply SIP response codes maps to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **local-response-map**—Enter the name of the configured SIP response map that you want to apply to this SIP interface for locally-generated SIP responses. This parameter is blank by default.

5. Save and activate your configuration.

Session Agent Ping Message Formatting

You can configure the user portions of the Request-URI and To: headers that define the destination of a session agent ping message, and the From: header that defines the source of a session agent ping message. These headers are sent to Oracle USM session agent. This feature is required for interoperability with certain E911 servers.

In the following example of a session agent ping-type message, you can set the user portion of the Request-URI (the text bob in the OPTIONS method line) and the user portion of the From: header (the text bob in the From: header) to the same new value. You can also set the user portion of the To: header (the text anna in the To: header) to its own new value.

```
OPTIONS sip:bob@sip.com SIP/2.0
From: UA1 <sip:bob@sip.com>
To: NUT <sip:anna@gw.sip.com>
Call-ID: 123abc@desk.sip.com
CSeq: 1 OPTIONS
Contact: <sip:UA1@client.sip.com>
```

```
Accept: application/sdp
Content-Length: 0
```

If you do not enable this feature, then the session agent ping-type message contains the text ping in all cases.

Session Agent Ping Message Formatting Configuration

1. Access the **session-agent** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)
   ```

2. Select the **session-agent** object to edit.

   ```
   ORACLE(session-agent)# select
   <hostname>:
   1: 192.168.100.101:1813

   selection: 1
   ORACLE(session-agent)#
   ```

3. **ping-from-user-part**—Set the user portion of the From: header that defines the source of a session agent ping message.

   ```
   ORACLE(session-agent)# ping-from-user-part bob
   ```

4. **ping-to-user-part**—Set the user portions of the Request-URI and To: headers that define the destination of a session agent ping message.

   ```
   ORACLE(session-agent)# ping-to-user-part anna
   ```

5. Type **done** to save your configuration.

SIP Statuses to Q.850 Reasons

This section explains the Oracle USM's ability to map Q.850 cause values with SIP responses, a feature used in SIP calls and calls that require IWF.

RFC 3326 defines a header that might be included in any in-dialogue request. This reason header includes cause values that are defined as either a SIP response code or ITU-T Q.850 cause values. You can configure the Oracle USM to support sending and receiving RFC 3326 in SIP messages for:

- Mapping H.323 Q.850 cause values to SIP responses with reason header and cause value
- Mapping SIP response messages and RFC 3326 reason header and cause
- Locally generated SIP response with RFC 3326 reason header and cause

As specified in RFC 3326, the Oracle USM sends SIP responses to the softswitch that contain the received Q.850 cause code and the reason.

Though the Oracle USM can generate RFC 3326 headers, the default behavior for this feature is disabled. Furthermore, the Oracle USM can receive and pass SIP error messages (4xx, 5xx, and 6xx) that contain the SIP reason header with a Q.850 cause code and reason (as specified in RFC 3326). If the system receives an error message without the Reason header, then the Oracle USM is not required to insert one.

In calls that require IWF, the Q.850 cause generated in the SIP response are the same as the cause received in the following H.225 messages: Disconnect, Progress, Release, Release Complete, Resume Reject, Status, and Suspend Reject. In addition, the Q.850 cause codes that the Oracle USM receives in RFC 3326 headers are passed to the H.323 part of the call unmodified; the H.323 call leg uses this cause code for releasing the call.

SIP-SIP Calls

The SIP Reason header might appear in any request within a dialog, in a CANCEL request, and in any response where the status code explicitly allows the presence of this header field. The syntax of the header follows the standard SIP parameter:

```
Reason: SIP;cause=200;text="completed elsewhere"
Reason: Q.850;cause=16;text="Terminated"
```

This feature attends to the following possible SIP call scenarios:

- When the Oracle USM receives a SIP request or SIP response that contains the Reason header, the Oracle USM passes it without modification.
- When it generates a SIP response, the Oracle USM includes the RFC 3326 Reason header containing a Q.850 cause code and reason. This is the case for all local conditions and for all internally generated error responses (4xx, 5xx, and 6xx) to an initial SIP INVITE. Possible local error scenarios are:
  - invalid-message
  - cpu-overloaded
  - media-released
  - media-not-allocated

SIP-SIP Calls Configuration

Configuring reason cause mapping for SIP-SIP calls requires that you set up the ACLI local-response-map configuration with appropriate entries; these generate the SIP response and the Q.850 cause code value to be used for particular error scenarios. If you want to add a Reason header, then you need to enable that capability in the global SIP configuration.

To configure a local response map:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **local-response-map** and press Enter.

   ```
   ORACLE(session-router)# local-response-map
   ORACLE(local-response-map)#
   ```

4. Type entries and press Enter.

   ```
   ORACLE(local-response-map)# entries
   ORACLE(local-response-map-entry)#
   ```

   From here, you can view the entire menu for the local response map entries configuration by typing a **?**.

5. **local-error**—Set the local error that triggers the use of this local response map; there is no default for this parameter. Valid values are:
   - **invalid-message**—Response map for invalid messages
   - **cpu-overload**—Response map for CPU overload
   - **enum-void-route**—Response map for when an ENUM server returns a ENUM +VOID response, or the local route table has 0.0.0.0 as the next hop
   - **media-released**—Response map for media release conditions
   - **media-not-allocated**—Response map for when media is not allocated
6. **sip-status**—Set the SIP response code to use. There is no default and the valid range is:
   - Minimum—100
   - Maximum—699
7. **sip-reason**—Set the SIP reason string you want to use for this mapping. There is no default value. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
8. **q850-cause**—Set the Q.850 cause. There is no default value.
9. **q850-reason**—Set the Q.850 reason string that you want to use for this mapping. There is no default value. If your value has spaces between characters, then your entry must be surrounded by quotation marks.

> **Note:**
>
> The following two parameters (method and register-response-expires) enable a SIP registration response mapping feature that allows you to configure the system to remap a SIP failure response—which it receives from another network device or that it generates locally—to a 200 OK. You might want the system to perform this type of mapping for circumstances where non-malicious endpoints continually attempt registration, but will stop (and still not be registered) when they receive a 200 OK. This response mapping does not actually register the client with the system, meaning that there is neither a registration cache entry or a CAM ACL for it.
>
> For the 200 OK it generates, the system removes any Reason or Retry-After header in the 200 OK and sets the expires time. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-After expires time). You can also set this value using the register-response-expires parameter, but the value you set should never exceed the Register request's expires time.

10. method—Enter the name of the received SIP failure response message you want to map to a 200 OK.

    There is no default for this parameter, and leaving the parameter empty turns off the SIP registration response mapping feature.
11. register-response-expires—Enter the time you want to use for the expires time what mapping the SIP method you identified in the method parameter from Step 4.

    The maximum is 999999999. By default, the expires time is the Retry-After time (if there is one in the response) or the expires value in the Register request (if there is no Retry-

After expires time). Any value you configure in this parameter (when not using the defaults) should never exceed the Register request's expires time.

12. Repeat this process to create the number of local response map entries that you need.
13. Save and activate your configuration for changes to take effect.

Adding the Reason Header

To enable the Oracle USM to add the Reason header:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type session-router and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type sip-config and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. add-reason-header—Enable this parameter to add the Reason header.

   The default value is disabled. The valid values are:

   - enabled | disabled

5. Save and activate your configuration for changes to take effect.

SIP Status

To configure a SIP status to Q.850 Reason with cause mapping:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-q850-map** and press Enter.

   ```
   ORACLE(session-router)# sip-q850-map
   ORACLE(sip-q850-map)#
   ```

4. Type **entries** and press Enter.

   ```
   ORACLE(sip-q850-map)# entries
   ORACLE(sip-q850-map-entry)#
   ```

   From here, you can view the entire menu for the SIP status to Q.850 Reason with cause mapping entries configuration by typing a **?**.

5. **sip-status**—Set the SIP response code that you want to map to a particular Q.850 cause code and reason. There is no default, and the valid range is:

   - Minimum—100
   - Maximum—699

6. **q850-cause**—Set the Q.850 cause code that you want to map to the SIP response code that you set in step 5. There is no default.

7. **q850-reason**—Set the Q.850 reason corresponding to the Q.850 cause code that you set in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
8. Repeat this process to create the number of local response map entries that you need.
9. Save and activate your configuration for changes to take effect.

   To configure a Q.850 cause to a SIP status with reason mapping:

10. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ```

11. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ```

12. Type **sip-q850-map** and press Enter.

    ```
    ORACLE(session-router)# q850-sip-map
    ORACLE(q850-sip-map)#
    ```

13. Type **entries** and press Enter.

    ```
    ORACLE(q850-sip-map)# entries
    ORACLE(q850-sip-map-entry)#
    ```

    From here, you can view the entire menu for the Q.850 cause to a SIP response code with reason mapping entries configuration by typing a **?**.

14. **q850-cause**—Set the Q.850 cause code that you want to map to a SIP status with reason. There is no default.
15. **sip-status**—Set the SIP response code to which you want to map the Q.850 cause that you set in step 5. There is no default, and the valid range is:
    - Minimum—100
    - Maximum—699
16. **sip-reason**—Set the reason that you want to use with the SIP response code that you specified in step 6. There is no default. If your value has spaces between characters, then your entry must be surrounded by quotation marks.
17. Repeat this process to create the number of local response map entries that you need.
18. Save and activate your configuration for changes to take effect.

    To enable the Oracle USM to add the Reason header for calls that require IWF:

19. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ```

20. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ```

21. Type **iwf-config** and press Enter.

    ```
    ORACLE(session-router)# iwf-config
    ACMEPACKET(iwf-config)#
    ```

22. **add-reason-header**—Enable this parameter to add the Reason header. The default is **disabled**. The valid values are:
    - enabled | disabled

23. Save and activate your configuration for changes to take effect.

Trunk Group URIs

The Oracle USM's trunk group URI feature, applicable for SIP and IWF signaling services, enables the capabilities related to trunk groups that are described in this section. This implementation follows the IPTEL draft Representing Trunk Groups in Tel/SIP Uniform Resource Identifiers (URIs) (draft-ietf-iptel-trunk-group-06.txt), and also supports more customized approaches.

- For a typical access call flow scenario, when the calling party's call arrives at the Oracle USM, the Oracle USM formulates a SIP INVITE message that it sends to a softswitch. The Oracle USM now supports a new URI contact parameter in the SIP request message so that service providers need to be able to:
  - Determine from where the Oracle USM received the call
  - Signal information about the originating gateway from a Oracle USM to a softswitch (e.g., an incoming trunk group or a SIP gateway to a Oracle USM)
- This feature supports the signaling of routing information to the Oracle USM from network routing elements like softswitches. This information tells the Oracle USM what egress route (or outgoing trunk groups) it should choose for terminating next hops/gateways. For this purpose, new SIP URI parameters in the Request-URI are defined. Additional URI parameters include the network context to identify the network in which the originating or terminating gateway resides.
- Especially important for large business applications, this feature can free Oracle USM resources by reducing the number of local policy, session agent, and session agent group configurations. By enabling the trunk group URI feature, the Oracle USM instead uses a routing scheme based on signaled SIP URI information.

Terminology

The following IPTEL terms are used in the descriptions of and instructions for how to configure this feature:

- Trunk—In a network, a communication path connecting two switching systems used in the establishment of an end-to-end connection; in selected applications, it may have both its terminations in the same switching system
- Trunk group—A set of trunks, traffic engineered as a unit, for the establishment of connections within or between switching systems in which all of the paths are interchangeable except where sub-grouped
- Trunk group name—Provides a unique identifier of the trunk group; referred to as tgrp
- Trunk group context—Imposes a namespace by specifying a domain where the trunk groups are; also referred to simply as context

Trunk Group URI Parameters

Trunk group URI parameters identify originating and terminating trunk group information in SIP requests.

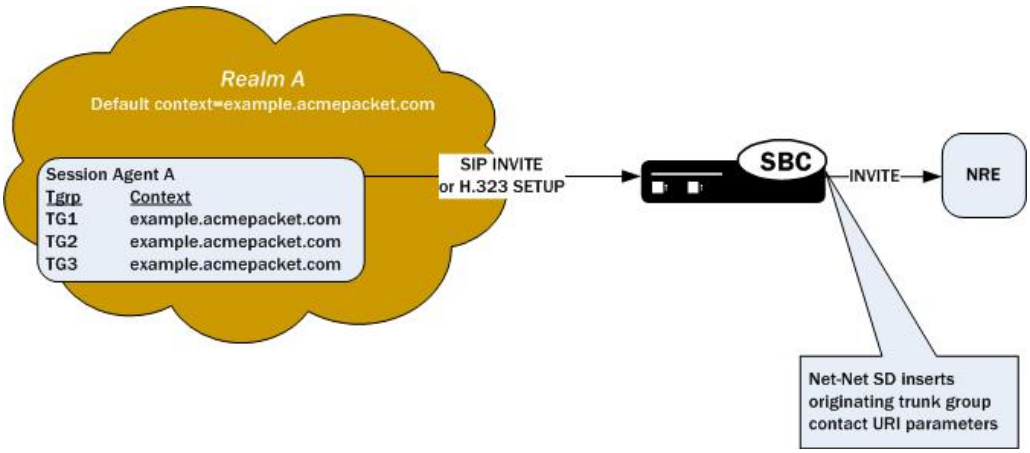
In the absence of official SIP standards for transporting trunk groups between signaling elements, the Oracle USM allows you to define URI parameters for use with originating and terminating trunk group URIs.

Originating Trunk Group URI Parameters and Formats

You can configure session agents and session agents groups on the Oracle USM to insert trunk group URI parameters in the SIP contact header. When SIP gateways comply with the IPTEL draft, they include the originating URI parameter in the SIP contact header. For those SIP and H.323 gateways that are not compliant, the Oracle USM inserts SIP trunk group URI parameters on the gateway's behalf.

When there are no applicable session agent or session agent group configurations, the Oracle USM uses the source IP address of the endpoint or gateway as the trunk group name (tgrp) parameter in the originating trunk group URI.

The following diagram shows a scenario where the Oracle USM inserts originating trunk group URI parameters.



There are two available formats for the originating trunk group URIs:

1. In compliance with the IPTEL draft, the first format has two parameters: tgrp (identifier of the specific trunk group) and trunk-context (defines the network domain of the trunk group). These appear in the following formats:
   - tgrp=”trunk group name”
   - trunk-context=”network domain”

     The URI BNF for would appear as it does in the example directly below, where the tgrp is tg55 and the trunk-context is trunk-context is telco.example.com:

     ```
     tel:+15555551212;tgrp=tg55;trunk-context=telco.example.com
     ```
2. The second format is customized specifically for access URIs and contains two provisioned parameters: tgrp (or tgname) and context (or provstring). This appears as tgrp.context, where these definitions apply:
   - tgrp (tgname)—Provisioned trunk group name for the originating session agent; this value must have at least one alphabetical character, cannot contain a period (.), and can contain a hyphen (-) but not as the first or the last character
   - context (provstring)—Name of the originating trunk group context; this value must have at least one alphabetical character in the top label

     This format conforms to format for a hostname in the SIP URI as specified in RFC 3261, such that a trunk group identifier would appear as:

```
custsite2NY-00020.type2.voip.carrier.net
```

where the tgrp is custsite2NY-00020, and the context is type2.voip.carrier.net.

The BNF for an access URI conforms to the following:

SIP-URI = "sip:" [userinfo] hostport uri-parameters [headers]

uri-parameters = *(";" uri-parameter)

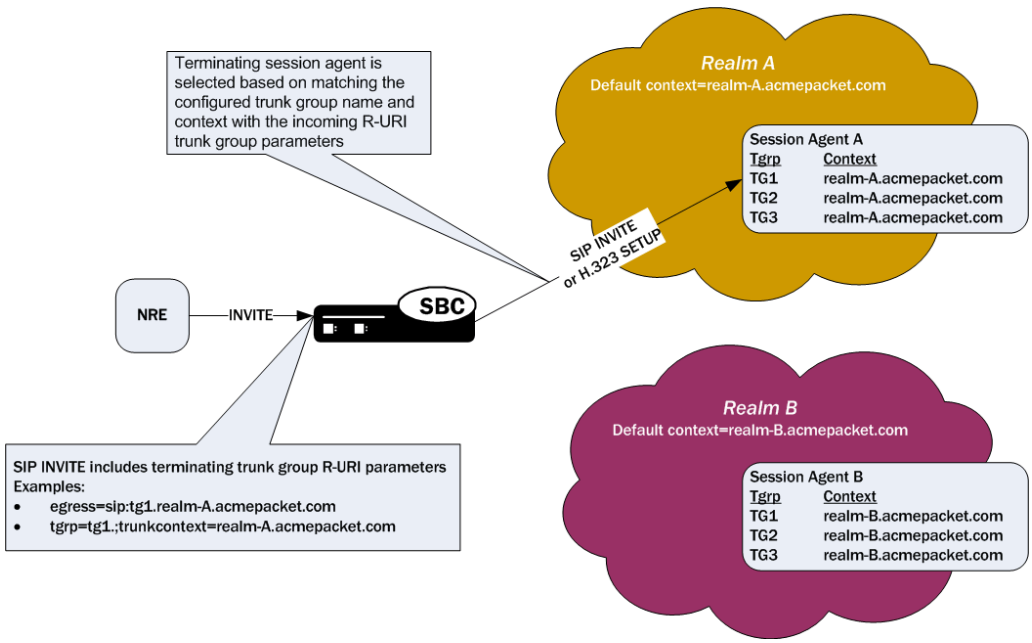
uri-parameter = transport-param / user-param / method-param

/ ttl-param / maddr-param / lr-param / other-param

```
other-param = accessid / pname [ '=' pvalue ]
accessid = "access=" accessURI
accessURI = scheme tgname ["." provstring]
scheme = "sip:" / token
tgname = ALPHA / *(alphanum) ALPHA *(alphanum / "-") alphanum /
    alphanum *(alphanum / "-") ALPHA *(alphanum) # up to 23 characters
provstring = *(domain ".") toplabel # up to 24 characters
toplabel = ALPHA / ALPHA *( alphanum / "-" ) alphanum
domain = alphanum/ alphanum *( alphanum / "-" ) alphanum
```

Terminating Trunk Group URI Parameters and Formats

Terminating trunk group URI parameters appear in the R-URI, and they can be included in by a network routing element to instruct the Oracle USM which egress trunk groups to use. By matching the trunk group URI parameter with configured session agents or session agent groups, the Oracle USM can locate the terminating gateway. The trunk group name can also be expressed as the IP address of the terminating gateway.



In the absence of official SIP standards for transporting trunk groups between signaling elements, the Oracle USM allows you to define the URI parameters used in terminating trunk groups.

There are two available formats for the terminating trunk group URIs:

1. In compliance with the IPTEL draft, the first format has two parameters: tgrp (which can be either a trunk group name or an IP address) and trunk-context (defines the network domain of the trunk group). These appear in the following formats:

   - tgrp=”trunk group name”
   - trunk-context=”network domain”

   An example R-URI with terminating trunk group parameters appears as follows, where the tgrp is TG2-1 and the context is isp.example.net@egwy.isp.example.net:

   ```
   INVITE sip:+15555551212;tgrp=TG2-1;trunk-
   context=isp.example.net@egwy.isp.example.net SIP/2.0
   ```

2. The second format is customized specifically for egress URIs and contains two provisioned parameters: tgrp (or tgname) and context (or tgdomain). This appears as tgrp.context (or tgname.tgdomain), where definitions apply:

   - tgrp (tgname)—Provisioned trunk group name for the originating session agent; this value must have at least one alphabetical character, cannot contain a period (.), and can contain a hyphen (-) but not as the first or the last character
   - context (tgdomain)—Name of the terminating trunk group context; this value can be up to twenty-four characters

   The use of multiple terminating trunk groups is not supported.

   The BNF for a single, egress URI with trunk group information conforms to:

   ```
   SIP-URI = "sip:" [userinfo ] hostport uri-parameters [headers ]
   uri-parameters = *( ";" uri-parameter )
   uri-parameter = transport-param / user-param / method-param
               / ttl-param / maddr-param / lr-param / other-param
   other-param = egressid  / pname [ '=' pvalue ]
   egressid = "egress=" egressURI
   egressURI = scheme tgname ["." tgdomain]
   scheme = "sip:" / token
   tgname = ALPHA / *(alphanum) ALPHA *(alphanum / "-") alphanum /
       alphanum *(alphanum / "-") ALPHA *(alphanum) # up to 23 characters
   tgdomain = *(domain ".") toplabel # up to 24 characters
   toplabel = ALPHA / ALPHA *( alphanum / "-" ) alphanum
   domain = alphanum/ alphanum *( alphanum / "-" ) alphanum
   ```

   For all trunk group URI support, you must set the appropriate parameters in the SIP manipulations configuration and in the session agent or session agent group configurations.

   In the originating trunk group URI scenario, a call arrives at the Oracle USM from a configured session agent or session agent group. If this session agent or session agent group has the appropriate trunk group URI parameters and inbound manipulation rules configured, the Oracle USM then looks to the SIP manipulations configuration and add the trunk group URI information according to those rules. Those rules tell the Oracle USM where and how to insert the trunk group URI information, and the system forwards the call.

   In the terminating trunk group scenario, a call arrives at the Oracle USM from, for instance, a call agent. This call contains information about what trunk group to use. If the information matches a session agent or session agent group that has outbound manipulation rules configured, the Oracle USM will then look up the SIP manipulations configuration and strip information according to those rules. Those rules tell the Oracle USM where and how to remove the information, and the Oracle USM forwards the call.

SIP Header and Parameter Manipulation

SIP header and parameter manipulation is its own configuration where you can set up rules for the addition, removal, and modification of a SIP header or the elements of a SIP header. For example, you can set up the configuration to add a URI parameter to the URI in a SIP header or replace an FQDN with in IP address. For trunk group URI support, this configuration tells the Oracle USM where and how to manipulate the SIP message to use originating (access) and terminating (egress) trunk group URI parameters.

These manipulations can be applied at the realm or at the session agent level.

Trunk Group Routing

You can configure SIP interfaces (using the ACLI **term-tgrp-mode** parameter) to perform routing based on the trunk group information received in SIP requests. There are three options: none, IPTEL, and egress URI.

- If you leave this parameter set to none (its default), the Oracle USM will not look for or route based on terminating trunk group URI parameters
- When you set this parameter to either **iptel** or **egress-uri** and the incoming request has the trunk group parameter of this type (IPTEL or egress URI), the Oracle USM will select the egress next hop by matching the “tgrp” and trunk context with a configured session agent or session agent group.
  If the received terminating trunk group URI parameters include an IP address, the egress next hop is the IP address specified. The Oracle USM determines the egress realm by matching the trunk context it receives with the trunk context you configure for the realm.
- If the incoming request does not have trunk group parameters or it does not have trunk group parameters of the type that you configure, the Oracle USM uses provisioned procedures and/or local policy for egress call routing.

The Oracle USM returns errors in these cases:

- If the terminating trunk group URI parameters do not identify a local Oracle USM session agent or session agent group, then the Oracle USM returns a SIP final response of 488 Not Acceptable Here.
- If the Oracle USM receives a SIP INVITE with terminating trunk group URI parameters that do not match the specified syntax, the Oracle USM returns a 400 final response with the reason phrase Bad Egress=Parameters.

Trunk Group URIs and SIP Registration Caching

For calls where SIP registration caching is used, you will need to set certain parameters that enable the Oracle USM to preserve trunk group URI parameters on the outgoing side.

- For SIP-SIP calls, you set the preserve-user-info **option** in the SIP interface configuration.
- For SIP-H.323 calls requiring IWF, you set the preserve-user-info-sa **option** in the session agent configuration.

Trunk Group URI Configuration

Before you configure your Oracle USM to support trunk group URIs, you need to determine:

- How you want to manipulate SIP headers (entered in the SIP header manipulations configuration)
- For terminating trunk group routing, the trunk group mode you want to use (none, IPTEL, or egress URI); this decides routing based on trunk group information
- The trunk group name and context to use entered in a session agent or session agent group configuration
- Whether you are using originating or terminating trunk group URIs (entered in the session agent configuration)
- The trunk group context for use in a realm configuration, in case the trunk group name in the session agent or session agent group does not have a context

Configuring SIP Manipulations

When you configure the SIP header manipulations to support trunk group URIs, take note of:

- The name of the configuration, so that you can use it when you apply the manipulations in a session agent for the inbound or outbound manipulations
- The **new-value** parameter, which specifies the trunk group and trunk group context that you want to manipulate; the possible values that apply to trunk group URI configurations are $TRUNK_GROUP and $TRUNK_GROUP_CONTEXT

Setting the Trunk Group URI Mode for Routing

To set the mode for routing for terminating trunk group URIs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **term-tgrp-mode**—Set the mode that you want to use for routing for terminating trunk group URIs. The default is **none**. Your choices are:
   - **none**—Disables routing based on trunk groups
   - **iptel**—Uses trunk group URI routing based on the IPTEL formats
   - **egress-uri**—Uses trunk group URI routing based on the egress URI format

Configuring a Session Agent for Trunk Group URIs

In a session agent, you can configure the outbound or inbound SIP header manipulation rules to use, as well as a list of trunk group names and contexts. For the trunk group names and contexts, you can use either the IPTEL or the custom format.

To configure a session agent for trunk group URIs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **out-manipulationid**—Enter the name of the SIP header manipulations configuration that you want to apply to the traffic exiting the Oracle USM via this session agent. There is no default.

5. **in-manipulationid**—Enter the name of the SIP header manipulations configuration that you want to apply to the traffic entering the Oracle USM via this session agent. There is no default.

6. **trunk-group**—In either IPTEL or custom format, enter the trunk group names and trunk group contexts to match. If you do not set the trunk group context, then the Oracle USM will use the one you set in the realm for this session agent.

   Your ACLI entries for this list must be one of these formats: tgrp:context or tgrp.context.

   To make multiple entries, surround your entries in parentheses and separate them from each other with spaces. For example:

   ```
   ORACLE(session-agent)# trunk-group (tgrp1:context1 tgrp2:context2)
   ```

7. **options**—If you want to configure trunk group URIs for SIP-H.323 calls that use the IWF and you are using SIP registration caching, you might need to add the preserve-user-info-sa to your list of session agent options.

   If you are adding this option to a new session agent, you can just type **options**, a Space, and **preserve-user-info-sa**.

   If are adding this to an existing session agent, you must type a plus (+) sign before the option or you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +preserve-user-info-sa.

Configuring a Session Agent Group for Trunk Group URIs

In a session agent group, you can configure the outbound or inbound SIP header manipulation rules to use, as well as a list of trunk group names and contexts. For the trunk group names and contexts, you can use either the IPTEL or the custom format.

To configure a session agent group for trunk group URIs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-group** and press Enter.

   ```
   ORACLE(session-router)# session-group
   ORACLE(session-agent-group)#
   ```

4. **trunk-group**—In either IPTEL or custom format, enter the trunk group names and trunk group contexts to match. If you do not set the trunk group context, then the Oracle USM will use the one you set in the realm for this session agent group.

Your ACLI entries for this list must take one of these formats: tgrp:context or tgrp.context.

To make multiple entries, surround your entries in parentheses and separate them from each other with spaces. For example:

```
ORACLE(session-agent-group)# trunk-group (tgrp1:context1 tgrp2:context2)
```

Setting a Trunk Group Context in a Realm

You can set trunk group contexts at the realm level, which will be used by all session agents and session agent groups if there is no context specified in their configurations.

The realm trunk group URI context accommodates the IPTEL and the custom format.

To configure a trunk group context for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **trunk-context**—Enter the trunk group context to use for this realm. There is no default.

Using this Feature with a SIP Interface

If you are using the trunk group URIs feature with SIP interface that has registration caching enabled, then you need to configure the preserve-user-info option for that SIP interface.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-group** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Add support for trunk group URIs with SIP interface that uses registration caching.

   If you are adding this option to a new SIP interface, you can just type **options**, a Space, and **preserve-user-info**.

   If are adding this to an existing SIP interface**, you must type a plus (+) sign before the option or you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign: options +preserve-user-info**.

Example 1 Adding Originating Trunk Group Parameters in IPTEL Format

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to add originating trunk parameters in IPTEL format.

```
sip-manipulation
        name                            add_iptel
        header-rule
                name                            contact
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            tgrp
                        type                            uri-user-param
                        action                          add
                        match-val-type                  any
                        match-value
                        new-value                       $TRUNK_GROUP
element-rule
                        name                            trunk-context
                        type                            uri-user-param
                        action                          add
                        match-val-type                  any
                        match-value
                        new-value                       $TRUNK_GROUP_CONTEXT
```

Example 2 Adding Originating Trunk Group Parameters in Custom Format

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to add originating trunk parameters in custom format.

```
sip-manipulation
        name                            add_att
        header-rule
                name                            contact
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            egressURI
                        type                            uri-param
                        action                          add
                        match-val-type                  any
                        match-value
                        new-value                       "sip:"+$TRUNK_GROUP+"."+
$TRUNK_GROUP_CONTEXT
```

Example 3 Removing IPTEL Trunk Group Names

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to remove IPTEL trunk groups names.

```
sip-manipulation
        name                            strip_iptel
```

```
        header-rule
                name                            request-uri
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            tgrp
                        type                            uri-user-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
element-rule
                        name                            trunk-context
                        type                            uri-user-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
```

Example 4 Removing Custom Trunk Group Names

This ACLI sample shows you how the ACLI SIP manipulations might appear in a case where you want to remove custom trunk groups names.

```
sip-manipulation
        name                            strip_egress
        header-rule
                name                            request-uri
                action                          manipulate
                match-value
                msg-type                        any
                element-rule
                        name                            egressURI
                        type                            uri-param
                        action                          delete-element
                        match-val-type                  any
                        match-value
                        new-value
```

Emergency Session Handling

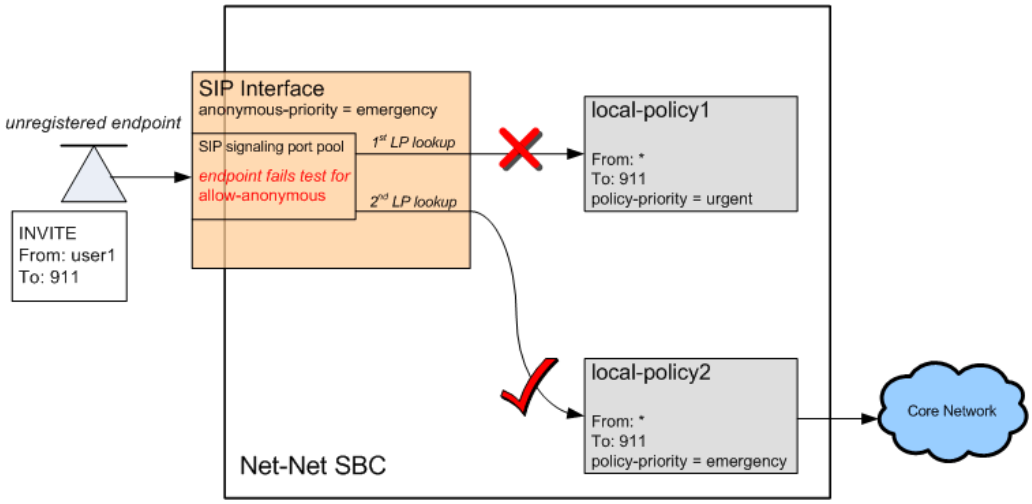
The Oracle USM provides a mechanism to handle emergency sessions from non-allowed endpoints. An endpoint is designated as non-allowed if it fails the admission control criteria specified by the allow-anonymous parameter in the SIP Ports configuration element.

When the Oracle USM receives a non-allowed emergency request, it performs a local policy lookup for a matching local policy. An emergency local policy could be configured to match if the To: header in a SIP message was addressed to 911.

An emergency policy priority selection criteria has been added to both the SIP interface and the local policy configuration elements. In the SIP interface, the parameter is called anonymous-priority. In the local policy, the parameter is called policy-priority.

For the Oracle USM to choose a local policy to route an emergency call, the emergency policy priority value on the local policy must be equal to or greater than the emergency policy priority value on the SIP interface where the emergency message was received. In this scheme, an emergency policy priority value of none is the lowest value and an emergency policy priority value of emergency is the highest.

When a match is made between all existing local policy criteria and the emergency policy priority, the emergency call will be sent to the core network according to the chosen local policy. In addition, the policy priority value of the chosen local policy is inserted into the Priority header of the core-bound SIP message..



Emergency Session Handling Configuration Procedures

Note the value of the allow-anonymous parameter in the SIP interface's SIP Ports for the incoming interface you are configuring. When an incoming emergency call from an unregistered endpoint can not be characterized by this setting, the Oracle USMwill use the following means to route the call.

Set the anonymous-priority parameter in the incoming SIP interface. This parameter specifies that for an INVITE received from an anonymous endpoint, the Oracle USM will choose a local policy of equal or greater policy priority for outbound routing.

Next, set the policy-priority parameter located in the local-policy configuration element. Most likely, this local policy will route messages to SIP devices that act on emergency calls. The local policy is selected when its value (or above) matches the anonymous-priority parameter in the sip-interface that receives the incoming phone call from an unregistered endpoint.

The enumerated values for both the anonymous-priority and policy-priority are: none, normal, non-urgent, urgent, emergency.

Emergency Session Handling Configuration

To set the anonymous priority for a message received in a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. Type **select** and the number of the SIP interface you want to configure.

   ```
   ORACLE(sip-interface)# select 1
   ```

5. **anonymous-priority—Set the policy priority for this SIP interface. It is used to facilitate emergency sessions from unregistered endpoints. This value is compared against the policy-priority parameter in the local-policy configuration element. The default is none. The valid values are:**

   - none | normal | non-urgent | urgent | emergency

     This completes the configuration.

     ```
     ORACLE(sip-interface)# anonymous-priority emergency
     ```

6. Save your work using the ACLI **done** command.

Setting Policy Priority

To set the policy priority for a local policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **local-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# local-policy
   ORACLE(local-policy)#
   ```

4. Type **select** and the number of the local policy you want to configure.

   ```
   ORACLE(local-policy)# select 1
   ```

5. **policy-priority—Enter the policy priority for this local policy. It is used to facilitate emergency sessions from unregistered endpoints. This value is compared against the anonymous-priority parameter in the sip-interface configuration element. The default is none. The valid values are:**

   - none | normal | non-urgent | urgent | emergency

     This completes the configuration.

     ```
     ORACLE(local-policy)# anonymous-priority emergency
     ```

6. Save your work using the ACLI **done** command.

Fraud Prevention

The Oracle USM can constrain outgoing SIP messages to a maximum size in bytes in order to support fraud prevention techniques. If a message does exceed the configured size, it is dropped. A SIP message can be constrained from 0 to 65535 bytes, with a default value of 4096 bytes.

Fraud Prevention Configuration

To set a maximum SIP message size:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. Type **select** to configure the existing sip config.

   ```
   ORACLE(sip-config)# select
   ```

5. **sip-message-len**—Set the size constraint in bytes of a SIP message. The default is **4096**. The valid range is:

   - Minimum—0
   - Maximum—65535

   This completes the configuration.

   ```
   ORACLE(sip-config)# sip-message-len 5000
   ```

6. Save your work using the ACLI **done** command.

SIP Early Media Suppression

This section explains how to configure SIP early media suppression, which lets you determine who can send early media and in what direction. Early media are the RTP/RTCP packets sent from the called party to the caller, or vice versa, before a session is fully established (before a 200 OK is received). When the Oracle USM receives an INVITE message with SDP, it can forward media packets to the calling endpoint as soon as it forwards the INVITE to the next hop. It can also forward media packets received from the calling endpoint to the called endpoint as soon as the Oracle USM receives SDP in a SIP response to the INVITE, usually a provisional message. This allows for any early media to be played, such as remote ringback or announcement.

Early media can be unidirectional or bidirectional, and can be generated by the caller, the callee, or both.

With early media suppression, you can block early media until the call is established. You can define which outbound realms or next hop session agents are allowed to send or receive early media. early media suppression only applies to RTP packets. RTCP packets received by Oracle USM are still forwarded to their destination in both directions, unless an endpoint is behind a NAT and the media manager has not been enabled for RTCP forwarding.

> **Note:**
>
> To use early media suppression, you cannot configure media release of any kind: same-realm, same-network, or multiple-system media release.

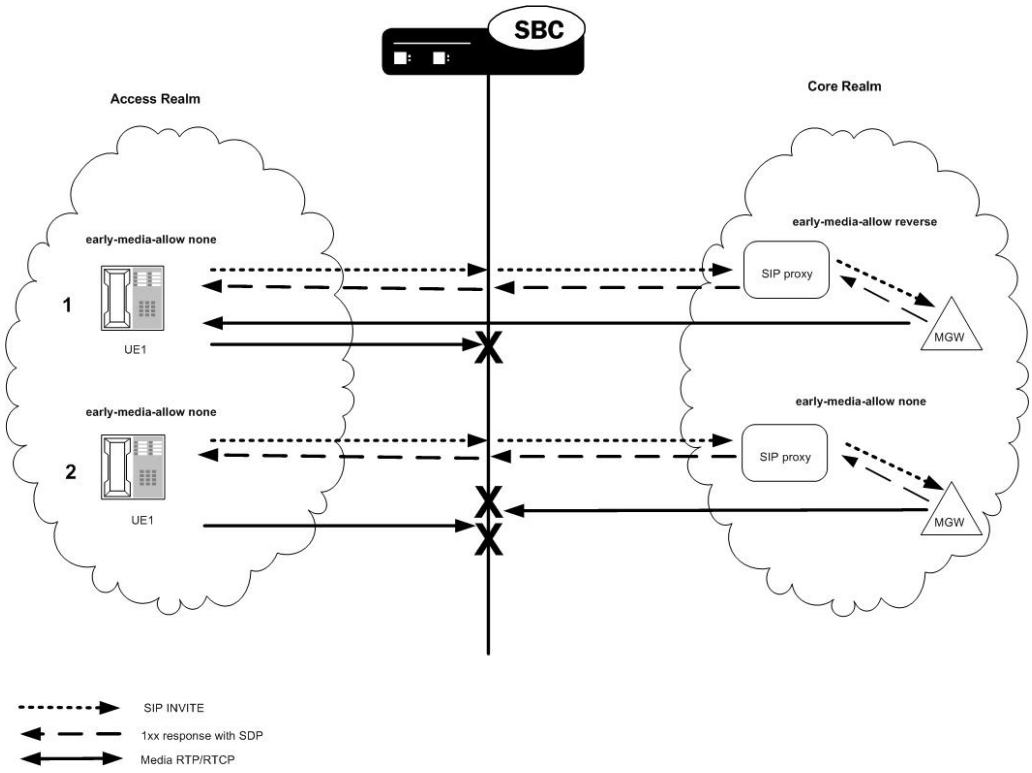
With the SIP-based addressing, early media suppression is based on the outbound SIP interface realms and the value of their early-media-allow parameter. When the Oracle USM forwards a SIP Invite out a SIP interface, the outbound realm is chosen based on the SIP layer information, such as the session agent for the next-hop or the address prefix of the next-hop SIP device. The matching realm's early-media-allow parameter value then applies to either allow all, block all, or block one-way early media until a 200 ok is received. At that point bidirectional media is allowed. The decision is based on SIP-layer addressing of next-hops.

You configure a rule for a realm or a session agent to use early media suppression. An early media suppression rule specifies whether you want to prevent early media in any direction, allow early media going to the calling endpoint in the reverse direction, or allow early media in both directions. The forward direction is when the packets flow from the caller to the called party. The reverse direction is when the packets flow from the called party to the caller.

The early media suppression rule is applied to a session. When the Oracle USM initiates a new session, it first checks whether the next hop is a session agent and if so, whether an early media suppression rule has been configured it. If an early media suppression rule is found, the Oracle USM enforces it. If the next hop is not a session agent or no early media suppression rule is configured, the Oracle USM checks whether an early media suppression rule has been configured for the outbound realm. If it finds one, it enforces it.

Example

The following illustration shows two examples of early media suppression.



1. Caller UE1 makes a call to the PSTN media gateway (MGW). The INVITE traverses from UE1 to the Oracle USM through the softswitch to the MGW. The Oracle USM allows early media from the core to reach UE1.
2. The PSTN MGW makes a call to UE1. The INVITE traverses to the Oracle USM and to UE1. The Oracle USM blocks all early media to and from UE1 until a 200 OK is received.

Early Media Suppression Support

The Oracle USM supports suppressing early media in the following directions no matter which side makes the SDP offer, until it receives 200 OK for an INVITE:

- Forward direction based on the outbound realm or next-hop session agent
- Forward and reverse directions based on the outbound realm or next-hop session agent.

The Oracle USM allows all media when a 200 OK response is received for the INVITE, regardless of whether the 200 OK response contains SDP.

Call Signaling

The Oracle USM media manager performs early media suppression according to an early media suppression rule. No change has been made to call signaling. For SIP, the Oracle USM still forwards SDP received in an INVITE request or response after performing a NAT to the media connection address. After which, the Oracle USM is ready to receive media packets from the endpoints. If an early media suppression rule has been configured, the Oracle USM drops the packets going in the direction being specified by the rule.

For a H.323 to SIP call, early media suppression rule does not change how the Oracle USM performs H.225/Q.931 call signaling and starts the H.245 procedure (if required) to establish logical channels for early media on the H.323 leg of the call.

Suppression Duration

When early media suppression is enabled in a session, the block lasts until the session is established. For a SIP to SIP call or an H.323 to SIP call, a session is established when the system receives a 200 OK response to the INVITE. A 200 OK response to the INVITE terminates early media suppression, even when it does not contain a SDP. (A 200 OK response to a PRACK or an UPDATE request does not terminate early media suppression.) After a session is established, the Oracle USM can receive a change in media session (for example, a re-INVITE with a new SDP) without an early media suppression rule blocking the media.

About the Early Media Suppression Rule

An early media suppression rule is configured in the form of a permission. It specifies whether early media is allowed in both directions, the reverse direction only or not at all. Reverse direction media is media sent in the upstream direction towards the calling endpoint.

Session Agent Rule

The next-hop session agent's early media suppression rule is applied regardless of whether the media packet's source or destination address is the same as the session agent's address. For example, if the session's next hop session agent is 10.10.10.5 but the SDP in a 183 response specifies 10.10.10.6 as its connection address.

Rule Resolution

When the call's next hop is a session agent and both the outbound realm of the call and the session agent have an early media suppression rule, the session agent's early media suppression rule takes precedence. If the session agent's early media suppression rule has not been configured, the outbound realm's early media suppression rule is used, if configured.

Selective Early Media Suppression

Normally, the Oracle USM performs early media blocking based on destination realm. Calls to such realms are prohibited from sending and receiving RTP until a SIP 200 OK response is received, and you can set the direction of the blocked media.

While decisions to block early media are customarily based on SIP-layer addressing, there are cases when the Oracle USM can reject early media based on the SDP address in the SDP answer for a 1XX or 2XX response. By comparing the SDP address with the realm prefix or additional prefix address, it can block early media for matching realms. For these cases, you define global or signaling realms—ones that are not tied to SIP interfaces, but which establish additional address prefixes and rules for blocking early media.

This way, the Oracle USM blocks all early media for SIP interface realms, but can accept it for global realms that reference media or PSTN gateways. This configuration allows early media for calls destined for the PSTN, and blocks it for user-to-user and PSTN-to-user calls.

Selective early media suppression addresses the fact that some service providers need to allow early media for certain user-to-user and PSTN-to-user calls to support, for example, custom ringback tones. The enhancements also address the fact that Oracle USMs can themselves lose

the ability to decide whether or not early media should be blocked when confronted with hairpinned call flows, or with traffic that traverses multiple Oracle USMs.

To address this need, you can configure realm groups. Realm groups are sets of source and destination realms that allow early media to flow in the direction you configure. For example, you can set up realm groups to allow media from PSTN realms to user realms so that users can listen to PSTN announcements, but prohibit early media from user realms to PSTN realms.

> **Note:**
>
> The source and destination realms you add to your lists need to be a global signaling realm matching the caller's SDP address prefix or a SIP realm.

Configuring the Realm

To configure the realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm
   ORACLE(realm)#
   ```

4. If configuring an existing realm, enter the select command to select the realm.
5. **early-media-allow**—Enter the early media suppression rule for the realm. The valid values are:
   - **none**—No early media is allowed in either direction
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle USM in the reverse direction is allowed

     There is no default value. If you leave this parameter blank, early media is allowed in either direction. You can use the following command to clear this parameter:

     ```
     early-media-allow ()
     ```

6. Save and activate your configuration.

   For example:

   ```
   realm-config
           identifier                      access1
           addr-prefix                     192.168.1.0/24
           network-interfaces
                                           media:0
           mm-in-realm                     enabled
           mm-in-network                   enabled
           msm-release                     disabled
           qos-enable                      disabled
           max-bandwidth                   0
           max-latency                     0
   ```

```
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        class-profile
        average-rate-limit              0
        access-control-trust-level      none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        deny-period                     30
        early-media-allow               none
        last-modified-date              2006-02-06 13:09:20
```

Configuring Session Agents

If you do not configure early media suppression for a session agent, the early media suppression for the outbound realm is used, if configured.

To configure session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. If configuring an existing session agent, enter the select command to select the session agent.

5. **early-media-allow**—Enter the early media suppression rule for the session agent. The valid values are:

   - **none**—No early media is allowed in either direction
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle USM in the reverse direction is allowed

   There is no default value. If you leave this parameter blank, early media is allowed in either direction. You can use the following command to clear this parameter:

   ```
   early-media-allow ()
   ```

6. Save and activate your configuration.

   For example:

   ```
   session-agent
           hostname                        cust1
           ip-address                      192.168.1.24
           port                            5060
           state                           enabled
           app-protocol                    SIP
   ```

```
app-type
transport-method                UDP
realm-id                        access1
description
carriers
allow-next-hop-lp               enabled
constraints                     disabled
max-sessions                    0
max-outbound-sessions           0
max-burst-rate                  0
max-sustain-rate                0
time-to-resume                  0
ttr-no-response                 0
in-service-period               0
burst-rate-window               0
sustain-rate-window             0
req-uri-carrier-mode            None
proxy-mode
redirect-action
loose-routing                   enabled
send-media-session              enabled
response-map
ping-method
ping-interval                   0
media-profiles
in-translationid
out-translationid
trust-me                        disabled
early-media-allow               reverse
last-modified-date              2006-05-06 13:26:34
```

Configuring Realm Groups

To configure a realm group for selective early media suppression:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-group** and press Enter.

   ```
   ORACLE(media-manager)# realm-group
   ORACLE(realm-group)#
   ```

4. **name**—Enter the name of the realm group.

5. **source-realm**—Enter the list of one or more global/SIP realms that you want to designate as source realms for the purpose of blocking early media; this is the realm identifier value for the realms you want on the list. Values in this list refer to calling SDP realms; this parameter has no default. To enter more than one realm in the list, list all items separated by a comma and enclose the entire entry in quotation marks:

   ```
   ORACLE(realm-group)# source-realm Private, Public
   ```

   To add a realm to the list, use the plus sign (+) in front of each new entry.

   ```
   ORACLE(realm-group)# source-realm +Private
   ```

You can also remove single items in the list by using the minus sign (-) directly in front of the realm identifier.

```
ORACLE(realm-group)# source-realm -Private
```

6. **destination-realm**—Enter the list of one or more global/SIP realms that you want to designate as destination realms for the purpose of blocking early media; this is the realm identifier value for the realms you want on the list. Values in this list refer to called SDP realms; this parameter has no default. To enter more than one realm in the list, list all items separated by a comma and enclose the entire entry in quotation marks:
7. ORACLE(realm-group)# **source-realm Private, Public**

   To add a realm to the list, use the plus sign (+) in front of each new entry.

   ```
   ORACLE(realm-group)# destination-realm +Private
   ```

   You can also remove single items in the list by using the minus sign (-) directly in front of the realm identifier.

   ```
   ORACLE(realm-group)# destination-realm -Private
   ```

8. **early-media-allow-direction**—Set the direction for which early media is allowed for this realm group. Valid values are:
   - none—Turns off the feature for this realm group by blocking early media
   - reverse—Allows early media to flow from called to caller
   - both (default)—Allows early media to flow to/from called and caller
9. Save and activate your configuration.

SDP-Response Early Media Suppression

This section explains how to configure SDP-response early media suppression, which can be used when the Oracle USM is deployed after a softswitch or proxy in the signaling path. In this deployment, user endpoints and gateways communicate directly with the softswitch or proxy, which in turn sends call signaling to the Oracle USM. The call signaling gets sent back to the same or different softswitch or proxy. Because the Oracle USM does not communicate with the endpoints or gateways that are the media terminators, early media suppression for this deployment must use SDP-based addressing rather than the SIP-based addressing (described in the SIP Early Media Suppression section in this technical notice).

Using this feature lets you configure specific IP addresses for which early media should not be suppressed, based on SDP addressing. The Oracle USM checks the SDP addresses in SIP responses against these IP address or address ranges to determine on which media gateway a call terminates.

SIP-Based Addressing

With the SIP-based addressing described in the SIP Early Media Suppression section, early media suppression is based on the outbound SIP interface realms and the value of their early-media-allow parameter. When the Oracle USM forwards a SIP Invite out a SIP interface, the outbound realm is chosen based on the SIP layer information, such as the session agent for the next-hop or the address prefix of the next-hop SIP device. The matching realm's early-media-allow parameter value then applies to either allow all, block all, or block one-way early media until a 200 ok is received. At that point bidirectional media is allowed. The decision is based on SIP-layer addressing of next-hops.

SDP-Based Addressing

SDP-response early media suppression follows the same sequence described for SIP-based addressing with one exception. A provisional response with SDP media can make the Oracle USM select a new early-media-allow rule from another realm, based on the addressing inside the responding SDP.

When the SDP-response early media suppression feature is enabled, the Oracle USM searches the outbound SIP interface's realms for a matching address prefix with the connection address in the responding SDP. If it finds a match, it uses the early-media-allow parameter value of that realm until the 200 OK message is received, then bidirectional media is allowed regardless. If the Oracle USM does not find a match, it searches all of the global realms for one. If it finds a match, the Oracle USM uses that realm's early-media-allow parameter value. If it does not find a match in the global realm(s), the Oracle USM continues to use the previous early-media-allow parameter value.

Global Realms

Global realms are realms that are not parents or children of any other realms, do not have defined SIP interfaces and ports (or any signaling interface or stack), and are configured to use the network interface lo0:0. They are special realms, applicable system-wide, and are currently only used for this feature. The only global realm configuration parameters applicable to early media suppression are:

- addr-prefix
- additional-prefixes
- early-media-allow
- network-interface (which must be set to lo0:0)

Additional Prefixes

You can specific additional prefixes in addition to that of the addr-prefix parameter you configure for a realm. For example, you can configure a global realm with additional address prefixes to specify the IP addresses (or ranges of addresses) of the media gateways that are allowed to send and receive early media. This overrides the SIP interface realm's early media blocking settings.

You can also enter additional prefixes in non-global realms. These additional prefixes function the same as would multiple values in the addr-prefix parameter (which only takes one value), except addresses in additional-prefixes are not used for SIP NATs.

Using the SDP-Response Early Media Suppression Rule

To use SDP-response early media suppression, you must add the early-media-sdp-realms option to the SIP interface configuration that interfaces with the next-hop device, such as the supported softswitch.

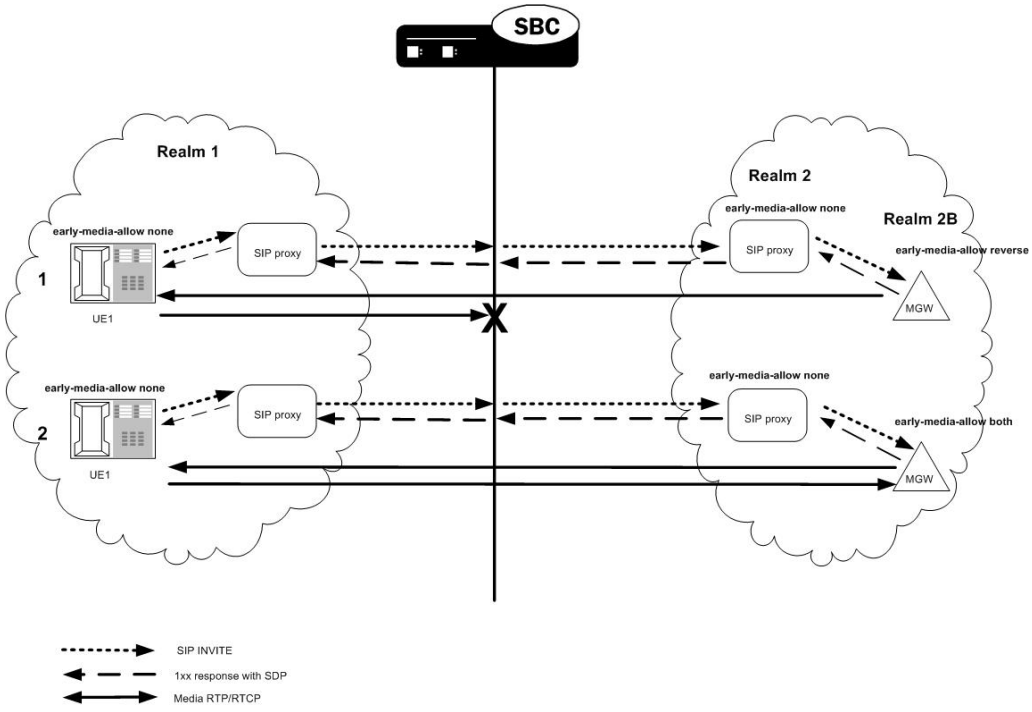
When the Oracle USM receives a provisional response that includes SDP from the called endpoint, and the early-media-sdp-realms option is active in the outgoing SIP interface of the call, it first searches the realms that apply to the outgoing SIP interface. If it does not find a realm, the Oracle USM searches the global realms. If the search yields a new realm that is not the SIP interface realm, its early media suppression rule (if any) replaces the existing one. Only the early media suppression rule of the new realm is applied to the call. Other realm properties

from the outbound realm remain applicable to the call. If no new realm is found, the early media policy of the outgoing SIP interface realm is applied.

TheOracle USM allows media when the SDP media connect address in a response matches one of a configured list of IP address ranges defined in a realm and the realm has early media allowed. You need to configure specific a IP address or address range to specify which media gateways should not be suppressed based on SDP media addresses. The IP addresses are checked against the SDP being received. The decision for suppression is based on whether the matching realm allows early media. The early media will be suppressed if the matching realm does not allow early media or if there is no match and the outbound SIP interface ream does not allow early media.

Example

The following illustration shows two examples of SDP-response early media suppression.



Configuring SDP-Response Early Media Suppression

To configure SDP-response early media suppression:

1. Add the early-media-sdp-realms option to the SIP interface that interfaces with the softswitch.
2. Configure the SIP interface realm with an early media suppression rule that blocks all early media.
3. Configure either or both of the following:
   - One or more of the SIP realm's child realms, each with an early media suppression rule that allows all or reverse direction early media and a list of additional prefixes that specifies the IP addresses of the media gateways, or a range of IP addresses that

includes the media gateways. Early media is allowed from these gateways only for calls that signals through this SIP interface.

- One or more realms that has the network interface equal to lo0:0, an early media suppression rule that allows all or reverse direction early media and a list of additional prefixes that specifies the IP addresses of the media gateways, or a range of IP addresses that includes the media gateways. Early media is allowed from these gateways regardless of interface.

Configuring the SIP Interface

To configure a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.

5. **options**—Enter early-media-sdp-realms as the option. If adding to an existing list of options, use a preceding plus (+) sign.

   ```
   options +early-media-sdp-realms
   ```

6. Continue to the next section to configure the outbound realm.

   For example:

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)# options + early-media-sdp-realms
   ORACLE(sip-interface)# done
   sip-interface
           state                         enabled
           realm-id                      access1
           sip-port
                   address                       192.168.1.30
                   port                          5060
                   transport-protocol            UDP
                   allow-anonymous               all
           carriers
           proxy-mode                    Proxy
           redirect-action
           contact-mode                  maddr
           nat-traversal                 none
           nat-interval                  30
           registration-caching          disabled
           min-reg-expire                300
           registration-interval         3600
           route-to-registrar            disabled
           teluri-scheme                 disabled
   ```

```
uri-fqdn-domain
options                         early-media-sdp-realms
trust-mode                      all
last-modified-date              2006-05-10 18:27:31
```

Configuring a Realm

To configure a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. If configuring an existing realm, enter the select command to select the realm.
5. **early-media-allow**—Enter the early media suppression rule for the realm. The valid values are:
   - **both**—Early media is allowed in both directions
   - **reverse**—Early media received by Oracle USM in the reverse direction is allowed
   - **none**—Early media is blocked
6. **additional-prefixes**—Enter a single or a comma-delimited list of IP address prefixes to use in addition to the value of the **addr-prefix** parameter.

   ```
   <IPv4> [/<number of bits>]
   ```

   <IPv4> is a valid IPv4 address and <number of bits> is the number of bits to use to match an IP address with the address prefix. Not specifying <number of bits> implies that all 32 bits are used for matching.

   Enclose the list between quotes if there is any space between a comma and the next address prefix.

   You can add and remove address prefixes to and from the list:

   - add-additional-prefixes adds one or more additional prefixes

     ```
     add-additional-prefixes 192.168.201.69
     ```

   - remove-additional-prefixes removes one or more additional prefixes

     ```
     remove-additional-prefixes 192.168.201.69
     ```

     If using multiple address prefixes, enter a comma-delimited list.

7. Save and activate your configuration.

   For example:

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)# additional-prefixes 192.168.200.0/24,192.168.201.68
   ORACLE(realm-config)# done
   ```

```
realm-config
        identifier                      early-media
        addr-prefix                     0.0.0.0
        network-interfaces
                                        media2:0
        mm-in-realm                     disabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        in-manipulationid
        out-manipulationid
        class-profile
        average-rate-limit              0
        access-control-trust-level      none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     30
        symmetric-latching              disabled
        pai-strip                       disabled
        trunk-context
        early-media-allow               reverse
        additional-prefixes             192.168.200.0/24
192.168.201.69
        last-modified-date              2006-05-11 06:47:31
```

SIP Duplicate SDP Suppression

Using the **strip-dup-sdp** option in the SIP configuration, you can enable your Oracle USM to suppress a duplicate SDP answer in the reliable responses (1xx and 2xx) in an INVITE transaction.

During INVITE transactions in certain networks, SDP answers in reliable provisional responses (1xx) can cause interoperability issues. This issue occurs when the UAS includes the SDP answer in subsequent 1xx responses or in the final 2xx response, and the UAC views that inclusion as a protocol violation. The UAC does so based on the fact that the SDP answer was reliably delivered to it in a 1xx response, and so it views additional SDP information as unnecessary in and ensuing reliable 1xx or 200 OK responses.

RFCs 3216 and 3262 do not specifically call out the UAS's including SDP information in this way as a protocol violation. Still, the system allows you set enable the **strip-dup-sdp** option as a means of preventing the UAC from terminating sessions. With this option enabled, the Oracle USM removes the SDP answer in subsequent reliable provisional or final 200 OK responses if it is identical to the SDP answer previously received.

SIP Duplicate SDP Suppression Configuration

To enable duplicate SDP suppression for SIP sessions:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing options, a Space, and then the option name **strip-dup-sdp**. Then press Enter.

   ```
   ORACLE(sip-config)# options +strip-dup-sdp
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

SIP SDP Address Correlation

SIP SDP address correlation ensures that when the Oracle USM receives a request containing SDP, the L3 source address of the request is compared against the address in the c-line of the SDP. When the addresses match, the session proceeds as it normally would. If there is a mismatch, the call is rejected with the default 488 status code. You can also configure the code you want to use instead of 488.

This functionality works only with non-HNT users. The value c=0.0.0.0 is an exception and is always processed.

SIP SDP Address Correlation Configuration Address Checking

The **sdp-address-check**, in the **enforcement-profile** element can be set to enable the SDP address correlation.

To enable SDP address checking:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **enforcement-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# enforcement-profile
   ORACLE(enforcement-profile)#
   ```

4. Use the ACLI **select** command so that you can work with the enforcement profile configuration to which you want to add this parameter.

```
ORACLE(enforcement-profile) select
```

5. **sdp-address-check**—Enable or disable SDP address checking on the Oracle USM. The default for this parameter is **disabled**.

   ```
   ORACLE(enforcement-profile)# sdp-address-check enabled
   ```

6. Save and activate your configuration.

   If a mismatch occurs and you want to reject the call with a status code other than 488, you set the code you want to use in the local response code map entries.

SIP SDP Address Correlation Configuration Mismatch Status Code

To apply a new status code to a SDP address correlation mismatch:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **local-response-map** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# local-response-map
   ORACLE(local-response-map)#
   ```

4. Type **entries** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(local-response-map)# entries
   ORACLE(local-response-map-entry)#
   ```

   From here, you can view the entire menu for the local response map entries configuration by typing a **?**.

5. **local-error**—Enter sdp-address-mismatch for which to apply the new status code.
6. **sip-status**—Set the SIP response code to use.
7. **sip-reason**—Set the SIP reason string you want to use for this mapping.

   ```
   ACMEPACKET(local-response-map-entry)# local-error sdp-addressmismatch
   ACMEPACKET(local-response-map-entry)# sip-status 403
   ACMEPACKET(local-response-map-entry)# sip-reason sdp address mismatch
   ```

8. Save and activate your configuration.

SIP SDP Address Correlation Configuration Enforcement Profile

To apply an enforcement profile to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm.

   ```
   ORACLE(realm-config)# enforcement-profile profile1
   ```

5. Save and activate your configuration.

SDP Insertion for (Re)INVITEs

If your network contains some SIP endpoints that do not send SDP in ReINVITEs but also contains others that refuse INVITEs without SDP, this feature can facilitate communication between the two types. The Oracle USM can insert SDP into outgoing INVITE messages when the corresponding, incoming INVITE does not contain SDP.

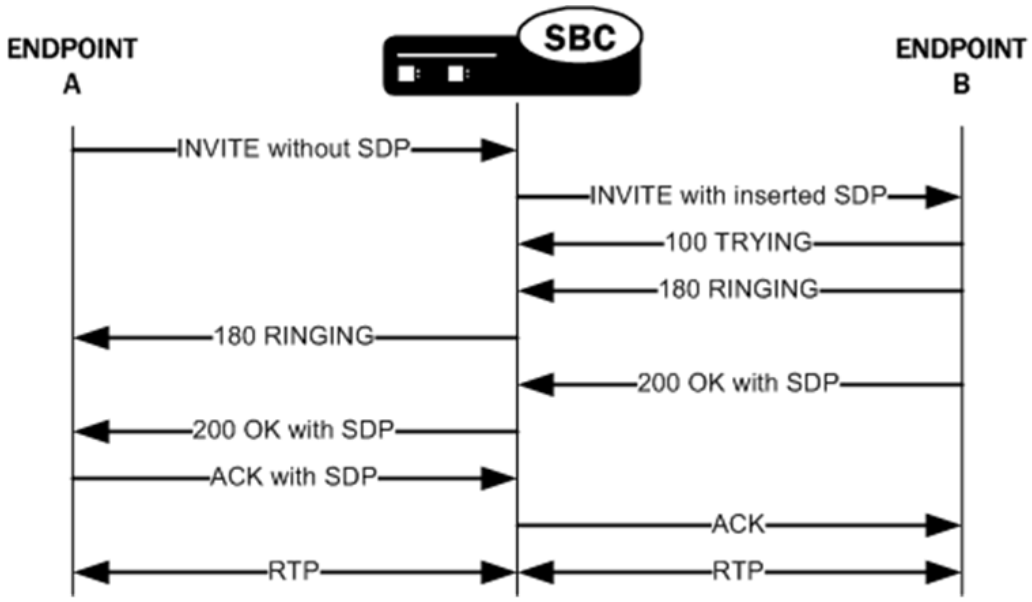
You can also use this feature when the network devices used in H.323-SIP interworking do not include SDP in the INVITEs sent to SIP endpoints. In this case, the Oracle USM can insert SDP in the outgoing INVITE messages it forwards to the next hop.

This feature works for both INVITEs and ReINVITEs.

This section explains how the SDP insertion feature works for INVITEs and ReINVITEs. The examples used this section are both pure SIP calls. Even when you want to use this feature for IWF calls, though, you configure it for the SIP side.

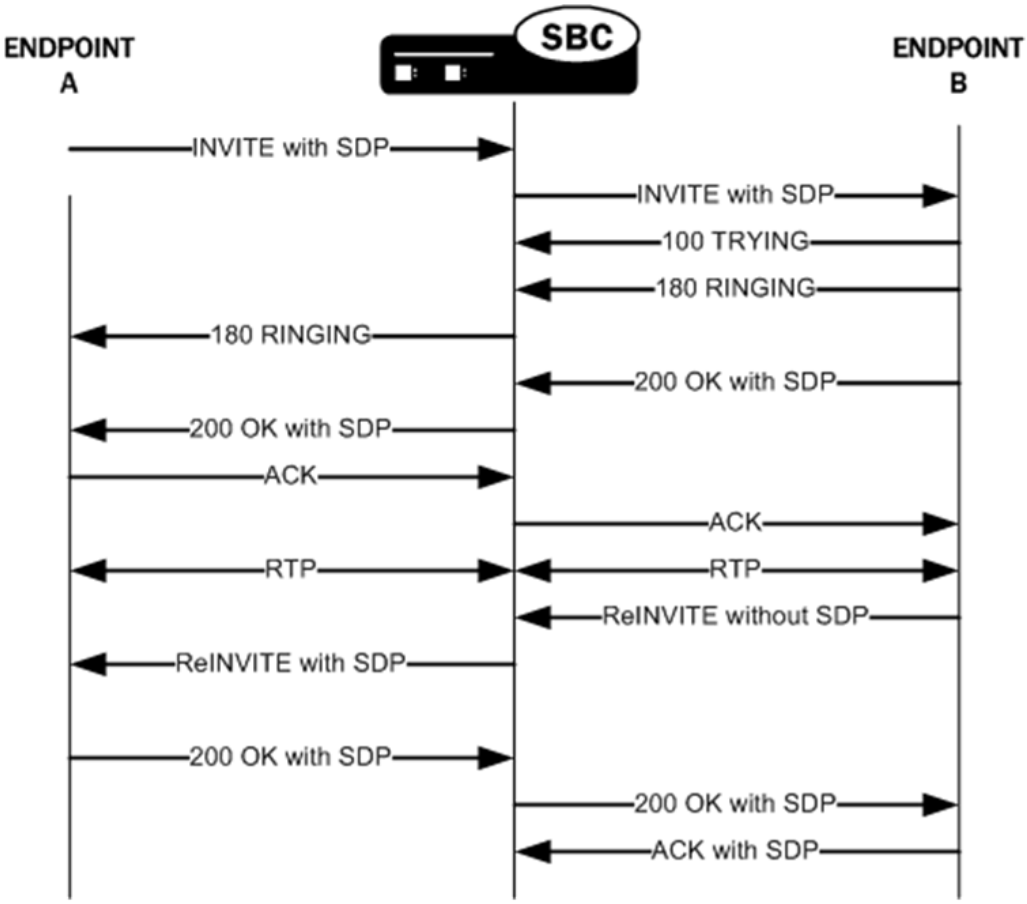
SDP Insertion for SIP INVITES

With the parameters mentioned above appropriately configured, the Oracle USM inserts SDP into an outgoing INVITE when the corresponding incoming INVITE has none. Because no SDP information is available for the session, the Oracle USM uses a media profile from a list of them you configure and then apply for SDP insertion.



SDP Insertion for SIP ReINVITEs

The section explains SDP insertion for ReINVITEs, using a case where SIP session has been established with an initial INVITE containing SDP. In the diagram below, you can see the initial INVITE results in a negotiated media stream. But after the media stream is established, Endpoint B sends a ReINVITE without SDP to the Oracle USM. In this case, the Oracle USM inserts the negotiated media information from the initial INVITE as the ReINVITE's SDP offer. For subsequent ReINVITEs with no SDP, the Oracle USM inserts the negotiated media information from the last successful negotiation as the ReINVITE's SDP offer. It then sends this ReINVITE with inserted SDP to the next hop signaling entity.



SDP Insertion Configuration

This section shows you how to configure SDP insertion for the calls cases described above.

Configuring SDP Insertion for SIP INVITEs

To work properly, SDP insertion for SIP invites requires you to set a valid media profile configuration.

To enable SDP insertion for INVITEs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-config)#
   ```

4. **add-sdp-invite**—Change this parameter from disabled (default), and set it to **invite**.

5. **add-sdp-profile**—Enter a list of one or more media profile configurations you want to use when the system inserts SDP into incoming INVITEs that have no SDP. The media profile contains media information the Oracle USM inserts in outgoing INVITE.

   This parameter is empty by default.

6. Save and activate your configuration.

Configuring SDP Insertion for SIP ReINVITEs

In this scenario, the Oracle USM uses the media information negotiated early in the session to insert after it receives an incoming ReINVITE without SDP. The Oracle USM then sends the ReINVITE with inserted SDP to the next hop signaling entity. You do not need the media profiles setting for ReINVITEs.

To enable SDP insertion for ReINVITEs:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-config)#
   ```

4. **add-sdp-invite**—Change this parameter from disabled (default), and set it to **reinvite**.
5. Save and activate your configuration.

SDP Version Change without SDP Body Change

When an SRTP call is made through the Oracle USM and a UE sends a reINVITE, there may be no change in the SDP contents. When the other UE responds, some devices will increment the o= line's session version, despite no SDP change. Normally the change in SDP version indicates that the SDP has changed, which would otherwise require the Oracle USM to modify the media flow set-up. In order to leave the media flows unchanged when session version changes but the rest of the SDP does not, an option is configured in the media-manager-config.

If the Oracle USM attempts to modify these media flows when unnecessary, calls could be dropped by the system disrupting media packet sequence number synchronization.

The ignore-reinv-session-ver option is set to handle this situation. When configured, the Oracle USM compares the current SDP received from a UE against the previously-received SDP from the same UE. If the SDP in the newer and older messages is the same, the Oracle USM ignores any changes to the session-version and will not modify the media flow portion of the call.

> **Note:**
>
> When the SDP change is only a reordering of SDP lines without any other change, the option has no effect.

This option is used only to mitigate compatibility issues when running SRTP calls over any of Acme Packet's IPSec accelerated NIUs for the Acme Packet 4500. NOT for ETC NIUs.

SDP Version Change Configuration

To configure media over TCP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring media over TCP.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **options** — Set the options parameter by typing **options**, a Space, the option name **ignore-reinv-session-ver** with a "plus" sign in front of it, and then press Enter.

   ```
   ORACLE(media-manager-config)# options +ignore-reinv-session-ver
   ```

   If you type the option without the "plus" sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a "plus" sign as shown in the previous example.

   Save and activate your configuration.

Restricted Media Latching

This section explains how to configure restricted media latching, which lets the Oracle USM latch only to media from a known source IP address, in order to learn and latch the dynamic UDP port number. The restricting IP address's origin can be either the SDP information or the SIP message's Layer 3 (L3) IP address, depending on the configuration.

About Latching

Latching is when the Oracle USM listens for the first RTP packet from any source address/port for the destination address/port of the Oracle USM. The destination address/port is allocated dynamically and sent in the SDP. After it receives a RTP packet for that allocated destination address/port, the Oracle USM only allows subsequent RTP packets from that same source address/port for that particular Oracle USM destination address/port. Latching does not imply that the latched source address/port is used for the destination of the reverse direction RTP packet flow (it does not imply the Oracle USM will perform symmetric RTP).

Restricted Latching

The Oracle Communications Session Border Controller restricts latching of RTP/RTCP media for all calls within a realm. It latches to media based on one of the following:

- SDP: the IP address and address range based on the received SDP c= connect address line in the offer and answer.
- Layer 3: the IP address and address range based on the received L3 IP address of the offer or answer. This option is for access registered HNT endpoints. If the L3 IP address is

locally known and cached by the Oracle USM as the public SIP contact address, that information could be used instead of waiting for a response. The Oracle USM might use the L3 IP address restriction method for all calls regardless of whether the endpoint is behind a NAT or not, for the same realms.

Symmetric Latching

A mode where a device's source address/ports for the RTP/RTCP it sends to the Oracle USM that are latched, are then used for the destination of RTP/RTCP sent to the device.

After allocating the media session in SIP, the Oracle USM sets the restriction mode and the restriction mask for the calling side as well as for the called side. It sets the source address and address prefix bits in the flow. It also parses and loads the source flow address into the MIBOCO messages. After receiving the calling SDP, the Oracle USM sets the source address (address and address prefix) in the appropriate flow (the flow going from calling side to the called side). After receiving the SDP from the called side, the Oracle USM sets the source address in the flow going from the called side to the calling side.

The Oracle USM uses either the address provided in the SDP or the layer 3 signaling address for latching. You also configure the Oracle USM to enable latching so that when it receives the source flow address, it sets the address and prefix in the NAT flow. When the NAT entry is installed, all the values are set correctly. In addition, sipd sends the information for both the incoming and outgoing flows. After receiving SDP from the called side sipd, the Oracle USM sends information for both flows to the MBCD so that the correct NAT entries are installed.

Enabling restricted latching may make the Oracle USM wait for a SIP/SDP response before latching, if the answerer is in a restricted latching realm. This is necessary because the Oracle USM does not usually know what to restrict latching to until the media endpoint is reached. The only exception could be when the endpoint's contact/IP is cached.

Relationship to Symmetric Latching

The current forced HNT symmetric latching feature lets the Oracle USM assume devices are behind NATs, regardless of their signaled IP/SIP/SDP layer addresses. The Oracle USM latches on any received RTP destined for the specific IP address/port of the Oracle USM for the call, and uses the latched source address/port for the reverse flow destination information.

If both restricted latching and symmetric latching are enabled, the Oracle USM only latches if the source matches the restriction, and the reverse flow will only go to the address/port latched to, and thus the reverse flow will only go to an address of the same restriction.

- Symmetric latching is enabled.
  If symmetric latching is enabled, the Oracle USM sends the media in the opposite direction to the same IP and port, after it latches to the source address of the media packet.
- Symmetric latching is disabled.
  If symmetric latching is disabled, the Oracle USM only latches the incoming source. The destination of the media in the reverse direction is controlled by the SDP address.

Example 1

A typical example is when the Oracle USM performs HNT and non-HNT registration access for endpoints. Possibly the SDP might not be correct, specifically if the device is behind a NAT. Therefore the Oracle USM needs to learn the address for which to restrict the media latching, based on the L3 IP address. If the endpoint is not behind a NAT, then the SDP could be used instead if preferred. However, one can make some assumptions that access-type cases will

require registration caching, and the cached fixed contact (the public FW address) could be used instead of waiting for any SDP response.

Example 2

Another example is when a VoIP service is provided using symmetric-latching. A B2BUA/proxy sits between HNT endpoints and the Oracle USM, and calls do not appear to be behind NATs from the Oracle USM's perspective. The Oracle USM's primary role, other than securing softswitches and media gateways, is to provide symmetric latching so that HNT media will work from the endpoints.

To ensure the Oracle USM's latching mechanism is restricted to the media from the endpoints when the SIP Via and Contact headers are the B2BUA/proxy addresses and not the endpoints', the endpoint's real (public) IP address in the SDP of the offer/answer is used. The B2BUA/proxy corrects the c= line of SDP to that of the endpoints' public FW address.

The Oracle USM would then restrict the latching to the address in the SDP of the offer from the access realm (for inbound calls) or the SDP answer (for outbound calls).

Restricted Latching Configuration

To configure restricted latching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Select the realm where you want to apply this feature.

   ```
   ORACLE(realm-config)# select
   identifier:
   1: Acme_Realm <none>           0.0.0.0
   2: H323REALM  <none>           0.0.0.0
   selection:1
   ORACLE(realm-config)#
   ```

5. **restricted-latching**—Enter the restricted latching mode. The default is **none**. The valid values are:

   - **none**—No restricted-latching used
   - **sdp**—Use the address provided in the SDP for latching
   - **peer-ip**—Use the layer 3 signaling address for latching

6. **restriction-mask**—Enter the number of address bits you want used for the source latched address. This field will be used only if the restricted-latching is used. The default is **32**; if this parameter uses this value, the complete IP address is matched. The valid range is:

   - Minimum—1
   - Maximum—32

7. Save your work using the ACLI **done** and **save** commands.

   The following example shows the realm configuration.

```
realm-config
        identifier                      Acme_Realm
        addr-prefix                     0.0.0.0
        network-interfaces
                                        public:0
        mm-in-realm                     enabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        class-profile
        average-rate-limit              0
        access-control-trust-level
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        deny-period                     30
        symmetric-latching              disabled
        pai-strip                       enabled
        mm-in-system                    enabled
        restricted-latching             sdp
        restriction-mask                30
        last-modified-date              2006-05-20 12:49:43
```

Enhanced SIP Port Mapping

This section explains how to configure SIP port mapping feature to support:

- Anonymous requests from endpoints
- Cases where endpoints dynamically change transport protocols between UDP and TCP

Anonymous Requests

If a SIP endpoint sends an INVITE message with a From header that is anonymous, the Oracle USM can find the registration cache entry by using the Contact and Via headers. In cases such as instant messaging (IM), where there is no Contact header, the Oracle USM can use the Via header.

The Oracle USM's checks whether the reg-via-key option is configured for the access-side SIP interface where a REGISTER is received. If the option is enabled, the Oracle USM makes the via-key by adding the IP address from the Via header to the firewall address (if there is a firewall present between the Oracle USM and the endpoint).

When an INVITE arrives at a SIP interface where this option is enabled, the Oracle USM determines whether the From header is anonymous or not. If it is anonymous, then the Oracle USMuses the Via-key to find the registration entry.

Anonymous SIP Requests Configuration

To enable support for anonymous SIP requests:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **options +reg-via-key** and press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-key
   ```

   If you type **options reg-via-key** without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Registration Via Proxy

The Oracle USM supports a number of features that require it to cache registration information for UAs (endpoints) registering and receiving requests through it. For those features to operate correctly, the Oracle USM must act as the outbound proxy through which these endpoints register.

In order to support deployments where a proxy sits between the Oracle USM and the endpoints, the Oracle USM must consider the bottom Via header it receives from endpoints when constructing and matching cache registration entries. And when you use SIP port mapping, the system must use the bottom Via header as a way to determine the endpoint uniquely so that it can have a unique mapping port when the SIP interface is configured with the reg-via-key=all option.

Using the reg-via-proxy option, you can enable your Oracle USM to support endpoints that register using an intervening proxy. You can set this option either for a realm or for a SIP interface. If you use it for a SIP interface, add to the SIP interface pointing toward the proxy and endpoints—the access side.

Considerations for Reg-Via-Key and Port Mapping

When you set the **reg-via-proxy** option, the Oracle USM includes the bottom Via header from received requests in the registration cache Via Key. The system also uses it for determining whether or not the request matches a registration cache entry. Each unique bottom Via received a unique mapping port when you turn SIP port mapping on and set the SIP interface with the **reg-via-key=all** option.

Request Routing

So that requests addressed to the corresponding registered contact are routed to the proxy, the Oracle USM includes the intervening proxy (i.e., the top Via) in the routing information for the registration cache when you set **reg-via-proxy**. To carry out this routing scheme, the system adds a Path header (if none is present) to the REGISTER. But it removes the Path header prior to sending the REGISTER to the registrar.

Note that when the received REGISTER contains a Path header, the Oracle USM uses it for routing requests toward the endpoint and includes it in the forwarded REGISTER request—as is the case when you do not enable SIP registration via proxy.

SIP Registration Via Proxy Configuration

To configure SIP registration via proxy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **options +reg-via-proxy** and press Enter.

   ```
   ORACLE(sip-interface)# options +reg-via-proxy
   ```

   If you type options reg-via-proxy without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

Dynamic Transport Protocol Change

The Oracle USM also uses the IP address and port in the Contact and Via headers. This is useful for cases when endpoints dynamically change transport protocols (TCP/UDP), and the port number used for sending an INVITE might not be the same one used to send a Register message.

If you do not enable this feature, when an endpoint registered with the Oracle USM used UDP for its transport protocol, a call fails if that endpoint subsequently initiates the call using TCP. The Oracle USM checks for the Layer 3 IP address and port, and it rejects the call if the port is changed.

With the new option reg-no-port-match added to the SIP interface configuration, the Oracle USM will not check the Layer 3 port in the INVITE and REGISTER messages.

Dynamic Transport Protocol Change Configuration

To enable dynamic transport protocol change:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **options +reg-no-port-match** and press Enter.

   ```
   ORACLE(sip-interface)# options +reg-no-port-match
   ```

   If you type options reg-no-port-match without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Privacy Extensions

This section explains how you can configure privacy services to be applied only when the source is trusted and the destination is considered untrusted. (Prior to this release, the Oracle USM always applied the privacy services, unless the source and the destination were both trusted.)

The Oracle USM considers all user endpoints and nodes outside the core as untrusted.

The Oracle USM acts as the boundary device between the trusted platform and the untrusted Internet, to implement privacy requirements. When it receives a message, the Oracle USM checks whether the source is trusted. It evaluates the level of privacy requested in a Privacy header, if present.

Depending on whether the source is trusted or untrusted, the Oracle USM can do different things when passing the message to the outgoing side. It also checks whether the destination is trusted.

Privacy Types Supported

The Oracle USM supports the following Privacy types:

- user: user-level privacy function provided. Any non-essential informational headers are removed, including the Subject, Call-Info, Organization, User-Agent, Reply-To, and In-Reply-To. Possibly the original value of the From header is changed to anonymous.
- header: headers that cannot be set arbitrarily by the user (Contact/Via) are modified. No unnecessary headers that might reveal personal information about the originator of the request are added. (The values modified must be recoverable when further messages in the dialog need to be routed to the originator.)

- id: third-party asserted identity kept private with respect to SIP entities outside the trust domain with which the user authenticated.

The following SIP headers can directly or indirectly reveal identity information about the originator of a message: From, Contact, Reply-To, Via, Call-Info, User-Agent, Organization, Server, Subject, Call-ID, In-Reply-To and Warning.

user

The Oracle USM supports the Privacy type user. It can remove non-essential information headers that reveal user information by:

- Setting the SIP From header and display information to anonymous
- Removing the Privacy header
- Removing Proxy-Require option tag = privacy (if present)
- Removing the following headers:
  Subject

  Call-Info

  Organization

  User-Agent

  Reply-To

  In-Reply-To

header

The Oracle USM also supports the Privacy type header. It modifies SIP headers that might reveal the user identity by:

- Stripping the Via header
- Replacing the Contact header
- Stripping Record-Route
- Removing the Privacy header
- Removing Proxy-Require option tag = privacy (if present)

In general, the B2BUA behavior of the Oracle USM by default provides header privacy for all sessions.

id

The Oracle USM also supports the Privacy type id. It keeps the Network Asserted Identity private from SIP entities outside the trusted domain by:

- Stripping only P-Asserted-Identity
- Removing the Privacy header and Proxy-Require option-tag = privacy
- Setting the From header to anonymous (for the backward compatibility)

Examples

The following examples show the actions the Oracle USM performs depending on the source and target of the calls.

Calls from Untrusted Source to Trusted Target

When calls are from an untrusted source to a trusted target and PPI is included in the INVITE sent to IP border elements, the Oracle USM maps the PPI information to PAI in the outgoing INVITE to the trusted side (even if the Privacy header is set to id or to none). The Privacy and From headers get passed on unchanged.

IP border elements must pass PAI (if received in the ingress INVITE) and the From and Privacy headers to the egress side just as they were received on the ingress side.

The Oracle USM maps the PPI to PAI by default, if the outgoing side is trusted. To change this behavior, you need to configure the disable-ppi-to-pai option.

Calls from Trusted to Untrusted

When calls are from a trusted source to an untrusted target, and the Privacy header is set to id, the Oracle USM strips PAI, makes the From header anonymous, and strips the Privacy header.

If the Privacy header is set to none, the Oracle USM does not change the From header and passes on the Privacy header, if there is one.

Calls from Trusted to Trusted

When calls are going from trusted source to trusted target acting as a peer network border element and PPI is included, the Oracle USM maps PPI to PAI. The Privacy header remains the same as signaled and the Oracle USM passes the From header and the PAI without changes.

Configuring SIP Privacy Extensions

Prior to this release the session agent's trust mode provided this functionality. Now you configure SIP interface's trust-mode as none, which means nothing is trusted for this SIP interface.

You also configure the disable-ppi-to-pai parameter disable the changing of the P-Preferred header to the P-Asserted-Identity header, if the outgoing side is trusted.

Trust Mode

To configure the trust mode:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. If configuring an existing interface, enter the select command to select the interface.
5. **trust-mode**—Select the trust mode for this SIP interface. The default value is **all**. The valid values are:
   - **all**—Trust all previous and next hops except untrusted session agents
   - **agents-only**—Trust only trusted session agents
   - **realm-prefix**—Trusted only trusted session agents or address matching realm prefix
   - **registered**—Trust only trusted session agents or registered endpoints
   - **none**—Trust nothing
6. Save and activate your configuration.

   The following example shows the trust-mode set to none. The remaining SIP interface options are omitted for brevity.

```
sip-interface
        state                           enabled
        realm-id                        access1
        sip-port
                address                         192.168.1.30
                port                            5060
                transport-protocol              UDP
                allow-anonymous                 all
        carriers
        proxy-mode                      Proxy
        redirect-action
        contact-mode                    maddr
        nat-traversal                   none
        nat-interval                    30
        registration-caching            disabled
        min-reg-expire                  300
        registration-interval           3600
        route-to-registrar              disabled
        teluri-scheme                   disabled
        uri-fqdn-domain
        options
        trust-mode                      none
```

Disabling the PPI to PAI Change

To disable the changing of PPI to PAI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

From this point, you can configure SIP configuration parameters. To view all sip-config parameters, enter a **?** at the system prompt.

4. If configuring an existing SIP configuration, enter the select command to select it.
5. **options**—Enter **disable-ppi-to-pai**. If adding to an existing list of options, use a preceding plus (+) sign.

   ```
   options +disable-ppi-to-pai
   ```

6. Save and activate your configuration.

SIP Registration Cache Limiting

Using SIP registration cache limiting for SIP endpoint access deployments, you can restrict the size of the SIP registration cache for the global SIP configuration.

You can implement this feature if you have been seeing issues where, either due to network failure scenarios or incorrect sizing of system capabilities, the Oracle USM and/or the SIP registrar cannot support the number of registering endpoints. Although the Oracle USM protects itself and the registrar against SIP REGISTER floods, conditions can still occur where too many legitimate endpoints attempt to register with the registrar via the Oracle USM.

By enabling SIP registration cache limiting, you restrict the number of legitimate endpoints that can register. The Oracle USM rejects any endpoints beyond the limit you set. If you do not want to use this feature, simply leave the reg-cache-limit parameter set to its default of 0, meaning there is no limit to the entries in the SIP registration cache.

When you limit the number of registered endpoints allowed in the Oracle USM's registration cache, the Oracle USM analyzes each registration before starting to process it. First, the Oracle USM checks the contact header to determine if it is already in the list of contacts for the user. If it finds the contact in its cache list, the Oracle USM treats the registration as a refresh; it treats any other headers as new. Note that the Oracle USM checks the message prior to making any changes to the cache because it must either accept or reject the message as a whole.

The Oracle USM adds the number of new contacts to the number already present in the cache, and rejects any registration with a contact that would cause it to exceed its limit. Rejection causes the Oracle USM to send a response communicating that its registration cache is full. The default response is the 503 Registration DB-Full message, but you can use the SIP response mapping feature to use another message if required.

You can set an option in the global SIP configuration that defines the value in the Retry-After header. The Oracle USM sends this header as part of its rejection response when the registration cache is full. Another option sets the percentage of the registration cache size which, if exceeded, causes the Oracle USM to send an alarm.

About Registration Cache Additions Modifications and Removals

When it receives a REGISTER message with new contact information for a user, the Oracle USM considers it an addition to the cache and augments the number of registration cache entries. Then the Oracle USM forwards the message to the registrar, and—when and only when the registrar returns both the original and new contacts in the 200 OK—the registration cache count stays the same. However, if the registrar returns only the new contact (making this a case of modification), then the Oracle USM removes the old contact information and subtracts accordingly from the number of registration cache entries.

Thus the Oracle USM does not know whether a REGISTER might result in an addition or a modification until it receives a response from the registrar. For this reason, the Oracle USM first assumes it is to make an addition, and then updates the registration cache and count when it has the necessary information from the registrar.

The registration cache count does not reflect removals during the rejection check because the Oracle USM ignores registration messages or expires headers with their expires values set to zero when it counts new entries. The fact that removals take place after additions and modifications means that messages which remove one contact while adding another might be rejected. That is, the addition might exceed the registration cache limit before any removal can take place to make room for it.

Registration Cache Alarm Threshold

A percentage of the registration cache limit, the registration cache alarm threshold is a configurable value you can set to trigger an alarm when the registration cache is reaching its limit. When exceeded, this threshold triggers the generation of an alarm and SNMP trap. When registrations fall back beneath the threshold, the Oracle USM clears the alarm and sends a clear trap.

This alarm is Major in severity, and its text reads as follows:

```
Number of contacts <registration count> has exceeded the registration cache
threshold <threshold %> of <registration cache limit value>.
```

Notes on Surrogate Registration

The Oracle USM does not, under any circumstances, reject surrogate registrations on the basis of the registration cache limit. However, surrogate registrations generate contacts, and so they do add to the global registration count. In the case where the surrogate registrations add to the registration count to the extent the count exceeds the limit you configure, you will have more registrations in the cache than the configured limit.

Monitoring Information

You can monitor how many entries are in the SIP registration cache using the ACLI **show registration** command and referring to the Local Contacts statistics.

SIP Registration Cache Limiting Configuration

This section shows you how to configure the registration cache limit, and how to set the options controlling retry times and thresholds for alarm purposes.

To configure SIP registration cache limiting:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

If you are adding this feature to an existing configuration, you need to select the configuration (using the ACLI **select** command) before making your changes.

4. **registration-cache-limit**—Set the registration cache limit, or the maximum number of SIP registrations that you want to keep in the registration cache. The minimum and default value for this parameter is 0, and you can set it to a maximum value of 999999999. Leaving this parameter set to 0 means there is no limit on the registration cache (and therefore leaves this feature disabled).
5. **options**—Set the options parameter by typing options, a Space, the option name **reg-cache-lim-retry-after=X** (where X is the value added to the Retry-After header) with a plus sign in front of it. This option defaults to 1800, and you can enter values from 0 to 999999999.

   You can configure the alarm threshold option the same way, substituting the option name **reg-cache-alarm-thresh=X** (where X is the percentage of registration cache limit that triggers an alarm). This option defaults to 95, and you can enter value from 0 to 100.

   ```
   ORACLE(sip-config)# options +reg-cache-lim-retry-after=2500
   ORACLE(sip-config)# options +reg-cache-alarm-thresh=90
   ```

   If you type options and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.
6. Save and activate your configuration.

SIP Registration Overload Protection

You can configure your Oracle USM for SIP Registration overload protection, which augments the Oracle USM's protection methods. Working with the Oracle USM's access control and registration caching functions, this new feature guards against benign avalanche restarts. The avalanche is caused by events where many endpoints lose power or connectivity at once, are restored to service, and then flood the Oracle USM as they attempt to register again.

Normally, the Oracle USM handles SIP registration by creating a temporary registration cache for the endpoint's address of record (AoR) and forwards the REGISTER request to the registrar. To challenge the endpoint's registration, the registrar sends back either a 401 Unauthorized or 407 Proxy Authorization Required response. When it receives the 401 or 407, the Oracle USM saves the challenge context in anticipation of receiving a second REGISTER with the endpoint's authentication credentials. The Oracle USM forwards the second REGISTER (with authentication credentials) to the registrar, and then the registrar confirms registration with a 200 OK. Both REGISTER requests are subject to the system's access control rules, set either for the ingress realm or the ingress session agent. The Oracle USM also honors the maximum registration sustain rate constraint for session agents; this applies when the incoming REGISTER is from a session agent and the outgoing REGISTER is sent to a session agent.

When you enable SIP Registration overload protection, the Oracle USM temporarily promotes the endpoint to the trusted level when it receives the 401 or 407 response (to the first REGISTER) from the registrar. This ensures that the second REGISTER (containing authentication credentials) can reach the Oracle USM. Temporary promotion lasts only for the amount of time remaining before the REGISTER server transaction expires plus the time allotted in the transaction expiration parameter in the SIP configuration. Before the temporary

promotion expires, there is enough time for any necessary retransmissions of the first REGISTER and for the second REGISTER to take place. The following situations might also occur:

- If the Oracle USM receives a 401 or 407 to the second REGISTER request, it resets its access control level for the endpoint's address to the default level; it then treats additional REGISTER requests from the same context at the default access control level.
- If the Oracle USM receives a 200 OK response to the REGISTER message, it extends the promotion time to the expiration period for the registration cache.

If the Oracle USM is able to find the temporary registration cache and the saved challenge context when the second REGISTER arrives, it forwards the REGISTER without checking the maximum registration sustain rate constraint for ingress and egress session agents—thereby ensuring that the REGISTER with authentication credentials is sent to the registrar. So when you use this feature, you should set the maximum registration sustain rate constraint of the session agent (representing the registrar) at half the registrar's maximum registration sustain rate. Additional REGISTER requests with the same challenge context are subject to the maximum registration sustain rate constraint.

SIP Registration Overload Protection Configuration

When you configure this feature, be sure to set the **reg-overload-protect** option in your global SIP configuration:

To enable SIP Registration overload protection on your Oracle USM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**reg-overload-protect**), and then press Enter.

   ```
   ORACLE(sip-config)# options +reg-overload-protect
   ```

   If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

> **Note:**
>
> Note that the sip-config option "cache-challenges" (enabled by default) must not have been disabled for SIP Registration Overload Protection to work properly. If you have disabled cache-challenges, re-evaluate the reason you disabled it. If registration overload protection supersedes your reason for disabling cache-challenges, re-enable the option as shown below.

```
ACMEPACKET(sip-config)# options +cache-challenges=yes
```

Note that the configuration syntax above is equivalent to the following, which uses the "-" character to remove the option.

```
ACMEPACKET(sip-config)# options -cache-challenges
```

5. Save and activate your configuration.

SIP Instance ID in Registration Cache

As defined in RFC 5626, Managing Client-Initiated Connections in the Session Initiation Protocol (SIP), the +sip.instance uniquely identifies a specific user agent instance. The instance-id does not change even when the User Agent is rebooted or power cycled. Using the instance-id from the +sip-instance parameter allows the SBC to locate a registered contact even when the IP address of the UA has changed.

The +sip.instance is a Contact header field parameter that contains a globally unique identifier, generally a Universally Unique Identifier (UUID) Uniform Resource Name (URN) that identifies a specific user agent client.

As defined in RFC 5626, Managing Client-Initiated Connections in the Session Initiation Protocol (SIP), the parameter contains an "instance-id" that uniquely identifies a specific user Agent instance. The instance-id does not change even when the User Agent is rebooted or power cycled (see Section 3.1 of RFC 5626). Using the instance-id from the +sip-instance parameter allows the SBC to locate a registered contact even when the IP address of the UA has changed.

An example REGISTER message containing a +sip-instance parameter and assigned value is shown below.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/TCP 192.0.2.2;branch=z9hG4bK-bad0ce-11-1036
Max-Forwards: 70
From: Bob <sip:bob@example.com>;tag=d879h76
To: Bob <sip:bob@example.com>
Call-ID: 8921348ju72je840.204
CSeq: 1 REGISTER
Supported: path, outbound
Contact: <sip:line1@192.0.2.2;transport=tcp>; reg-id=1;
;+sip.instance="<urn:uuid:00000000-0000-1000-8000-000A95A0E128>"
Content-Length: 0
```

The user agent calculates a +sip.instance generally using a time-stamp, a unique string -- such as a MAC address, and/or a randomly generated number. After calculating the value, the user agent saves it to persistent storage, thus ensuring that the value is unchanged by subsequent power cycles.

SIP Instance ID and the Registration Cache

Every assignment of a new IP address to a mobile user agent client results in a new REGISTER request. The receipt of the REGISTER request, in turn, generates an additional entry in the registration cache. The implementation can be simplified by including the +sip.instance value in the criteria used to match incoming REGISTER requests with existing sessions maintained in the registration cache. When a match is found, the SBC re-uses the existing registration cache rather than adding a new one.

If such a +sip.instance match is found and the contact address has changed, the SBC determines if digest authentication has been used in previous associated REGISTER requests. If so, the SBCs forwards the REGISTER request to the registrar for authentication of the new location.

SIP Instance ID Configuration

Because this configuration is dynamically performed at the **sip-config** level, it applies to all SIP interfaces and takes effect immediately

1. Access the **sip-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **match-sip-instance**—Set this parameter to enabled to use the +sip-instance-id when matching incoming calls with the registration cache. Valid values are:
   - **disabled**—(the default) disables the use of the +sip-instance-id when matching incoming calls with the registration cache
   - **enabled**—enables the use of the +sip-instance-id when matching incoming calls with the registration cache

```
ACMEPACKET(sip-config)# match-sip-instance enabled
ACMEPACKET(sip-config)#
```

4. Type **done** to save your configuration.

SIP Request Method Throttling

You can configure throttling mechanisms for SIP INVITEs and REGISTERs using session agent constraints. However, you might want to throttle other types of SIP methods, and for those methods you should use the rate constraints configuration available both in the session constraints (which you then apply to a SIP interface or a realm) and the session agent configurations.

Oracle recommends you use session agent constraints for session-rate INVITE throttling and registration-rate for REGISTER throttling.

For SIP access deployments, you can configure rate constraints for individual method types along with a set of burst and sustain rates. These constraints can help to avoid overloading the core network. In addition, they restrain the load non-INVITE messages use, thus reserving capacity for INVITE-based sessions and Registrations

When you configure SIP request method throttling, you must exercise care because it is possible to reject in-dialog requests. Therefore, Oracle recommends you do NOT configure constraints—although the configuration allows you to and will not produce error messages or warnings if you set them—for the following SIP method types:

- ACK
- PRACK
- BYE
- INFO
- REFER

However, the Oracle USM is likely to throttle NOTIFY requests despite their being part of a Subscribe dialog.

Therefore, the methods you will most likely configure for throttling are:

- NOTIFY
- OPTIONS
- MESSAGE
- PUBLISH
- REGISTER

The Oracle USM counts Re-INVITEs and challenged responses against the throttle limit, but does not check to determine if the constraints have been exceeded for either.

You can configure separate constraints—inbound and outbound values for burst and sustain rates—for each different method type you configure. Although you should use session agent constraints (and not rate constraints) for INVITEs, if you also set up rate constraints for INVITEs, then the smallest configured value takes precedence.

About Counters and Statistics

Each rate constraint you configure for a SIP method tracks its own counters. For example, if you configure a rate constraint for the PUBLISH method, the burst and sustain rates you set for it apply only to the PUBLISH method and not to any other methods for which you might set up rate constraints. You can, however, set the burst rate window in the session constraints configuration that will apply to all methods configured as rate constraints.

The Oracle USM captures statistics for SIP methods throttled by rate constraints for SIP interfaces and session agents; it does not capture these statistics for the global SIP configuration.

SIP Request Method Throttling Configuration

This section shows you how to set up rate constraints for session constraints (which are then applied to SIP interfaces) and session agents.

To use this feature, you must enable the **extra-method-stats** parameter in the global SIP configuration.

To set the extra-method-stats parameter in the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
If you are adding this feature to an existing configuration, you need to
select the configuration (using the ACLI select command) before making your
changes.
```

4. **extra-method-stats**—Set this parameter to **enabled**.
5. Save and activate your configuration.

Rate Constraints for SIP Interfaces

To apply rate constraints to SIP interfaces, you need to configure rate constraints in the session constraints configuration and then apply the session constraints to the SIP interface where you want them used.

Note that you need to set up the parent **session-constraint** configuration to save any rate constraints you configure.

To configure rate constraints:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-constraints** and press Enter.

```
ORACLE(session-router)# session-constraints
ORACLE(session-constraints)#
```

   If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

4. Type **rate-constraints** and press Enter.

```
ORACLE(session-constraints)# rate-constraints
ORACLE(rate-constraints)#
```

5. **method**—Enter the SIP method name for the method you want to throttle. Although the parameter accepts other values, your entries should come only from the from the following list for the feature to function properly:
   - NOTIFY
   - OPTIONS
   - MESSAGE
   - PUBLISH
   - REGISTER
6. **max-inbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.

7. **max-outbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
8. **max-inbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
9. **max-outbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
10. Save your changes and apply this session constraint and its rate constraint(s) to SIP interfaces.

Applying Session and Rate Constraints to a SIP Interface

You need the name of the session constraints configuration to apply the restrictions you set up to a SIP interface.

To apply session and rate constraints to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **constraint-name**—Enter the name of the session constraint configuration where you have set up rate constraints to apply them to this SIP interface. This parameter has no default, and must be the valid name of a session constraint configuration.
5. Save and activate your configuration.

Configuring Rate Constraints for Session Agents

You can also use this feature for individual SIP session agents.

To configure rate constraints for a SIP session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

If you are adding rate constraints to an existing configuration, then you will need to select the configuration you want to edit.

4. Type **rate-constraints** and press Enter.

```
ORACLE(session-agent)# rate-constraints
ORACLE(rate-constraints)#
```

5. **method**—Enter the SIP method name for the method you want to throttle. Your entries should come only from the following list:
   - NOTIFY
   - OPTIONS
   - MESSAGE
   - PUBLISH
   - REGISTER
6. **max-inbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
7. **max-outbound-burst-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound burst rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
8. **max-inbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the inbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
9. **max-outbound-sustain-rate**—For the SIP method you set in the methods parameter, enter the number to restrict the outbound sustain rate on the SIP interface where you apply these constraints. The default and minimum value is 0, and the maximum is 999999999.
10. Save and activate your configuration.

SIP Delayed Media Update

The Oracle USM supports SIP delayed media update. When enabled, this feature keeps the Oracle USM from updating its media flow information for flows established after an offer-answer exchange. The Oracle USM does not update the flow information until a new offer and answer arrive for a specific set of media flows.

The (subsequent) offer does not have to be for the same session; rather, it can appear as a new SIP INVITE that uses the same SDP.

Delayed Media Update Disabled

When this feature is disabled (which is the default behavior), the Oracle USM updates media flow entries in its CAM based on signaled SDP when it processes the SDP. If it processes an SDP offer, Oracle USM allocates steering port resources; the Oracle USM updates any missing elements for the flow when the answer is returned.

In cases when a secondary offer arrives (either a reINVITE, an UPDATE, or the original INVITE is hairpinned back through the Oracle USM), the Oracle USM updates the following media flow information at the time of the offer

- Destination IP address
- Destination port
- Realm for the media flows
- Media release settings

This behavior affects specific applications that are better served by the Oracle USM waiting to update media flow information until it receives the answer to the second offer.

Delayed Media Update Enabled

When you enable the SIP delayed media update feature, the Oracle USM:

- Delays changing the active media flow CAM entry for a new offer if a previous offer and answer have been received for the same media flows; it encodes new SDP information in an outgoing offer, but does not change the CAM entry until the answer is received
- Delays changing the active media flow CAM entry even when the new offer is for a new session
- Supports media release when performing delayed media update changes
- Offers per-realm configuration

This section describes how the delayed media update feature works for hairpinned call flows and for an SDP offer arriving for installed flows.

- Hairpinned call flows—In this type of call flow, the application server (AS) sends an INVITE back to the Oracle USM and that INVITE needs to be forwarded to another user (user B). When it receives the offer in this INVITE and delayed media update is disabled, the Oracle USM determines that the call is hairpinned and deletes the CAM entry for the flow for user A, who has sent the initial INVITE. The Oracle USM deletes the CAM entry for the flow from the AS to user A.
  With delayed media update enabled, the CAM entry for the flow from the AS to user A is not deleted. Instead, the Oracle USM waits until it has an answer from user B, and then performs the necessary updates and deletions.
- SDP offer for installed media flows—With delayed media update enabled, if it has received an offer and answer and a new offer arrives for the same flow, the Oracle USM delays updating the CAM entries until an answer is received for the new offer.

SIP Delayed Media Update Configuration

You enable this feature on a per-realm basis by setting one parameter.

To enable SIP delayed media update:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
```

If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **delay-media-update**—Enable keeping the Oracle USM from updating its media flow information for flows established after an offer/answer exchange. The default is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

Expedited Call Leg Release for Preempted Hairpin Calls

When hairpinned calls are ended because of signaling failures (such as a SIP mid-dialog signaling timeout, or an H.323 TCP keepalive failure) on one call leg, the Oracle USM deletes both legs' media flows simultaneously by default. In addition, when the first hairpinned call leg is torn down, the second call leg is gracefully released immediately by the Oracle USM creating and sending an appropriate signaling message (e.g., BYE for a SIP call or ReleaseComplete for an H.323 call) to the endpoint.

You can override this behavior by configuring the **dont-terminate-assoc-legs** option in the media manager. When configured, the orphaned call leg in the hairpin scenario will be torn down after the **initial guard timer expires.** The disconnect times of the two call legs, as recorded by the accounting appliation, will be significantly different, due to the initial guard time for the second call leg.

Accounting Considerations

To indicate cases like this, where the second leg of the hairpinned call was preempted, Oracle USM includes the following combination of release/termination causes in the CDR:

```
VSA 49: Acct-Terminate-Cause = NAS_REQUEST
VSA 62: Acme-Disconnect-Cause = 8
```

SIPconnect

The Oracle USM supports the SIPconnect model, wherein PBXs register themselves so that service providers do not need to know IP addresses or locations in advance for static configurations. This is particularly helpful when the PBX is behind a NAT.

In the PBX registration process, the PBX creates a binding between one of its phone numbers as the address of record (AoR) and Contact-URI in the REGISTER message. The registrar knows that the single AoR actually represents many addresses, and so it registers them implicitly. However, the registrar does not return the implicit AoR number in P-Associated-URIs.

The SIPconnect feature resolves the following issues that arise from using this model:

- SIP INVITEs sent to the PBX from the Registrar through the Oracle USM have the Request-URI of registered contact. Because it typically ignores the To-URI, the PBX needs the Request-URI username portion to be the specific extension number being called. With the SIP connect feature enabled, the Oracle USM overwrites the Request-URI username with the To-URI username.

- SIP INVITEs from the PBX have the From AoR and Contact-URI usernames of specific phones rather than of the registered AoR and Contact-URI. For the Oracle USM, this means that it cannot use the **allow-anonymous** parameter value of register; there would be no registered user matches, and the Oracle USM would reject them (with a 403 Forbidden).
  With the SIP connect feature enabled, the Oracle USM performs allow-anonymous checking based on the registered Via address, which is the same for all requests for the same PBX.

Modifications to Registration Caching Behavior

With the SIP connect feature enabled, Oracle USM registration caching works the same way that it does with the feature disabled, with the following exceptions:

The Oracle USM determines whether the destination realm has the sip-connect-pbx-reg option configured, and then:

- If it is configured, the Oracle USM replaces the user part of the Request-URI with the user part of the To header. When the INVITE contains a P-Called-Party-ID header, the Oracle USMuses the user part of the P-Called-Party-ID header (instead of the To header).
- If it is not configured, the Oracle USM determines if the destination address is for a session agent and whether that session agent has sip-connect-pbx-reg option configured. When it is configured, the Oracle USM performs the same replacements described in the bullet directly above. When it is not configured, the Oracle USM does not make any replacements.

When it receives an INVITE request, the Oracle USM checks the incoming realm for the sip-connect-pbx-reg option.

- If it is configured, the Oracle USMC uses the INVITE's source address (instead of the AoR and Contact-URI) to search the registration cache for a matched registration entry.
- If it is not configured, the Oracle USM determines if the INVITE's source address is for a session agent and whether that session agent has sip-connect-pbx-reg option configured. When it is configured, the Oracle USM replaces the user part of the Request-URI with the user part of the To header. When the INVITE contains a P-Called-Party-ID header, the Oracle USMuses the user part of the P-Called-Party-ID header (instead of the To header).

  When it is not configured, the Oracle USM does not make any replacements.

Configuring SIP Connect Support

You configure this feature by adding the sip-connect-pbx-reg option to the realm configuration. In addition, though this feature requires that your configuration also be set up as outlined in this section. The first two items are required, and Acme Packet recommends that you also implement the suggested additional configuration.

Required Configuration

- Registration caching is enabled.
- For the realm from which registrations come, the options list must include sip-connect-pbx-reg; this is new configuration introduced to support this feature. The presence of this option instructs the Oracle USM to skip matching the Contact header in the INVITE request with the registered Contact of the registration entry. The Oracle USM finds a registration using only the INVITE's source address.

Alternatively, you can configure the sip-connect-pbx-reg option in the options list for a session agent. When the realm where an INVITE comes from does not have this option set, the Oracle USM determines whether or not the INVITE came from a session agent. You might choose to configure session agents with this option if you do not want it applied to an entire realm. If the PBX is behind a NAT device, the session agent's IP address for the PBX (if statically configured) must be the IP address of the NAT device. And if DNS is use, the session agent's hostname must resolve to the NAT device's IP address.

Suggested Additional Configuration

- In the SIP ports configuration (accessed through the SIP interface configuration), the **allow-anonymous** parameter must be set to registered. This setting allows the Oracle USM to accept SIP requests from session agents and registered endpoints only, but to accept REGISTER requests from any endpoint.
- For the SIP interface that accepts registrations, the **options** parameter must be set to reg-via-key. This setting allows the Oracle USM to use the source address of an INVITE as the key to find a registration entry in the registration cache. When the INVITE's Contact header matches the registered Contact in the registration entry, the Oracle USM accepts the INVITE request.

SIP Connect Configuration

To set the SIP connect option for a realm configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **sip-connect-pbx-reg** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(realm-config)# options +sip-connect-pbx-reg
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

   To set the SIP connect option for a SIP session agent configuration:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   Oracle USM# configure terminal
   ```

7. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   Oracle USM(configure)# session-router
   ```

8. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

9. **options**—Set the options parameter by typing **options**, a Space, the option name **sip-connect-pbx-reg** with a plus sign in front of it, and then press Enter.

   ```
   Oracle USM(session-agent)# options +sip-connect-pbx-reg
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the session agent's configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

10. Save and activate your configuration.

SIP Registration Event Package Support

Certain endpoints subscribe to the Registration Event Package, RFC 3680, which defines how SIP user agents can request and obtain notifications about registration events. Previously, the Oracle USM passed the Subscribe and Notify messages of this package transparently, without modifying the XML bodies of either. However, in many cases the XML body can contain IP addresses, contact URIs, and expires times that the Oracle USM needs to modify for proper operation. This new feature enables the Oracle USM to modify correctly the XML body for the Registration Event Package.

In addition to resolving this type of issue, enabling registration event package support on your system provides the functions described below:

- The Oracle USM performs NAT on all contacts in the reginfo, regardless of their state.
- The Oracle USM performs NAT on the address of record (AoR) attribute of the Registration element when it matches an existing cache entry. When either the Contact-URI or the AoR does not match a cache entry and the host part of the URI is an IP address, the Oracle USM will NAT the host part using the applicable SIP NAT configuration
- Contacts are found in the XML URI element for the contact. But if there is no URI element, then the Oracle USM uses the Contact element information for the contact.
- If the expires attribute in the Contact element is a value other than zero, the Oracle USM uses (inserts) the expires values from the registration cache.
- This feature also introduces delayed deletion from the registry cache. When a 200 OK comes back in response to a REGISTER message and the 200 OK does not include all previously registered contacts, the missing contacts are deleted. If the global SIP configuration option contact_cache_linger=XX (where XX is the number of seconds to wait before deleting), then the contacts to be deleted remain for the specified number of seconds before they in fact are deleted.

Updating Expiration Values

This feature also supports updating the expiration values for the registration cache when a Contact element has the expires attribute. For this support, the following apply:

- If the value of the expires attribute is greater than the expiration value for the access-side registration cache entry, the Oracle USM replaces the XML expires attribute value with the cached one from the access side.
- If the value of the XML expires attribute is less than the core-side expiration value for the core-side registration cache entry, the Oracle USM updates the core-side expiration value with the value from the expires attribute. Further, the Oracle USM adjusts the access-side expiration value of the registration cache in these ways:
  - If the value of the XML expires attribute is less than the current access-side expiration value for the registration cache entry, the Oracle USM sets the access-side expiration value to be equal to the value in the expires attribute.
  - Otherwise, the Oracle USM leaves the expires value for the access-side expiration value for the registration cache entry unchanged. If this happens, the Oracle USM replaces the value of the XML expires attribute with the adjusted access-side expiration value.
- If the expires attribute from a Contact element is 0 (meaning that the core is removing the registration), the Oracle USM removes that Contact-URI from its registration cache. And if the registration cache entry has no remaining Contact-URIs, the Oracle USM deletes the registration cache entry altogether.

Contact Cache Linger Configuration

You enable this feature as part of the global SIP configuration, using that configuration's **options** parameter. You can optionally configure the number of seconds you want to keep a contact in the registration cache before it is deleted. This is the option:

- **contact-cache-linger=XX**—Number of seconds to wait before a contact is deleted from the cache (where XX is the number of seconds)
  To enable SIP Registration overload protection on your Oracle USM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**contact-cache-linger=XX**) where XX is the number of seconds to keep a contact in the cache before deleting it.

   ```
   ORACLE(sip-config)# options +contact-cache-linger=5
   ```

   If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

5. Save and activate your configuration.

SIP Event Package for Registrations

This feature enables the Oracle USM, acting as a Proxy Call Session Control Function (P-CSCF) to initiate subscription to the SIP Event Package for Registrations. Support for the SIP Event Package for Registrations requires no special license.

Applicable Standards

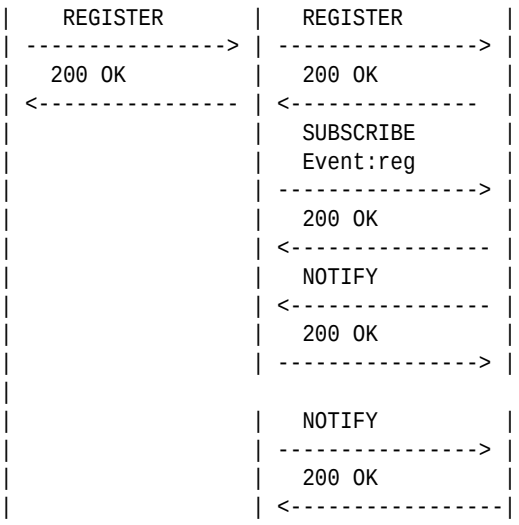
RFC 3265, *Session Initiation Protocol (SIP)-Specific Event Notification*, outlines a framework within which a SIP node, designated as the subscriber, can request automatic, asynchronous notification of certain events from a remote peer, designated as the notifier. RFC 3265 also defines an Event Package as a set of state information to be reported by a notifier to a subscriber, and mandates that such Event Packages be described in a specific RFC explicitly identifying the state information to be reported by the notifier and defining required syntax and semantics to support the subscription/notification exchange.

RFC 3680, *A Session Initiation Protocol (SIP) Event Package for Registrations*, fulfills this requirement by defining a method that enables SIP user agents to request a defined set of state information from a SIP Registrar.

Section 5.2.3 of the 3GPP (Third Generation Partnership Project) 24.229, *IP Multimedia Call Control Protocol Based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP)*; Stage 3, mandates that a P-CSCF subscribe to the SIP Event Package for Registrations as defined in RFC 3680.

Call Flow

An example call flow between a SIP endpoint, the Oracle USM, acting as a P-CSCF, and a SIP Registrar (S-CSCF) illustrates the Subscription/Notification process.



The first two messages (the REGISTER request and the 200 REGISTER response) accomplish the successful registration of the SIP endpoint.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP pc34.example.com;branch=z9hG4bKnaaff
From: sip:joe@example.com;tag=99a8s
To: sip:joe@example.com
```

```
Call-ID: 88askjda9@pc34.example.com
CSeq: 9976 REGISTER
Contact: sip:joe@pc34.example.com
```

Immediately after processing the initial 200 OK from the SIP Registrar, the Oracle USM sends a SUBSCRIBE Request to the S-CSCF.

```
SUBSCRIBE sip:joe@example.com SIP/2.0
Via: SIP/2.0/UDP app.example.com;branch=z9hG4bKnashds7
From: sip:sd.example.com;tag=123aa9
To: sip:joe@example.com
Call-ID: 9987@app.example.com
CSeq: 9887 SUBSCRIBE
Contact: sip:joe@pc34.example.com
P-Asserted-Identity: <sip:sd@example.com>
Event: reg
Max-Forwards: 70
Accept: application/reginfo+xml
```

The Request URI and To header contain the address-of-record (sip:joe@example.com) of the subscription subject. This value was previously contained in the From and To headers of the of the original REGISTER request. These fields are always identical in REGISTER requests, except in the case of third-party registration.

The From and P-Asserted-Identity headers contain the identity of the subscription requester.

The Contact header contains an IP address or FQDN at which the subscription subject can be reached. This field duplicates the value of the Contact header in the original REGISTER request. Multiple contacts can be registered for a single address-of-record.

The Event header contains the required value, reg, which identifies the requested Event Package subscription. reg specifies the SIP Event Package for Registrations.

The Accept header contains the required, default value, application/reginfo+xml, indicating syntactical support for Registration notifications and attached XML notification bodies.

Assuming the S-CSCF accepts the subscription, it responds with a 200 SUBSCRIBE response.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP app.example.com;branch=z9hG4bKnashds7
;received=192.0.2.1
From: sip:sd.example.com;tag=123aa9
To: sip:joe@example.com;tag=xyzygg
Call-ID: 9987@app.example.com
CSeq: 9987 SUBSCRIBE
Contact: sip: joe@pc34.example.com
Expires: 3600
```

The From header contains the identity of the subscription requester.

The To header contains the address-of-record of the subscription subject.

The Contact header contains an IP address or FQDN at which the subscription subject can be reached.

The Expires header contains the subscription duration in seconds. Upon receipt of a 2xx response to the SUBSCRIBE request, the Oracle USM stores the information for the established dialog and the expiration time. If continued subscription is required, the Oracle USM automatically refreshes the subscription to the SIP Event Package for Registrations, either 600 seconds before the expiration time if the initial subscription was for greater than

1200 seconds, or when half of the time has expired if the initial subscription was for 1200 seconds or less.

Following the 200 SUBSCRIBE response, the S-CSCF generates an initial notification, with Event Package state information contained in an XML body.

```
NOTIFY sip:app.example.com SIP/2.0
Via: SIP/2.0/UDP server19.example.com;branch=z9hG4bKnasaij
From: sip:app@example.com;tag=xyzygg
To: sip:sd.example.com;tag=123aa9
Call-ID: 9987@app.example.com
CSeq: 1289 NOTIFY
Contact: sip:server19.example.com
Event: reg
Max-Forwards: 70
Content-Type: application/reginfo+xml
Content-Length: ...

<?xml version="1.0"?>
<reginfo xmlns="urn:ietf:params:xml:ns:reginfo" version="1"
state="partial">
    <registration aor="sip:joe@example.com" id="a7" state="active">
        <contact id="76" state="init" event="registered"
        duration-registered="0">
            <uri>sip:joe@pc34.example.com</uri>
        </contact>
    </registration>
</reginfo>
```

Notification Bodies

Registration state changes are reported in XML attachments to NOTIFICATIONS generated by the S-CSCF. As shown above, the XML consists of one or more registration elements that report the state of a specific address-of-record. Attributes supported by the registration element are as follows:

aor contains the address-of-record

id identifies this specific registration

state reports the Registration state — init, active, or terminated

init — address-of-record not yet cached

active — address-of-record maintained in current cache

terminated — address-of-record removed from cache, not currently valid

registration elements, in turn, contain one or more child contact elements. Attributes supported by the contact element are as follows>

id identifies this specific contact

state reports the Contact state — active or terminated

event reports the event that generated the last state change — registered, created, refreshed, shortened, expired, deactivated, probation, unregistered, or rejected

duration-registered reports the length (in seconds) of the current registration

contact elements, contain a single child uri element that identifies the contact address of FQDN.

SIP Event Package for Registrations Configuration

Subscription to the SIP Event Package for Registrations is enabled at the SIP interface level.

1. Use the following command sequence to move to **sip-interface** Configuration Mode.

   ORACLE# **configure terminal**

   ORACLE(configure)# **session-router**

   ORACLE(session-router)# **sip-interface**

   ORACLE(sip-interface)#

2. Use the **subscribe-reg-event** parameter to subscribe to the SIP Event Package for Registrations.

   By default, subscription is disabled.

   ORACLE(sip-interface)# **subscribe-reg-event enabled**

   ORACLE(sip-interface)#

3. Use **done**, **exit**, and **verify-config** to complete enabling the Event Package subscription.

SIP Transport Selection

With this feature enabled, when the Oracle USM forwards a message larger than the value specified in the maximum UDP length parameter, it attempts to open on outgoing TCP connection to do so. This connection might fail for a number of reasons; for example, an endpoint might not support UDP, or it might be behind a firewall. The UDP fallback option addresses this condition. If it is configured in SIP interfaces associated with an outgoing message and a TCP session cannot be established, the Oracle USM falls back to UDP and transmits the message. When the option is not present, the Oracle USM's default behavior is to return the SIP status message 513 Message too Large.

SIP Transport Selection Configuration

You enable this feature per SIP interface by setting options that control the maximum UDP length and allow UDP fallback:

- max-udp-length=X (where X is the maximum length)—Sets the largest UDP packers that the Oracle USM will pass. Packets exceeding this length trigger the establishment of an outgoing TCP session to deliver the packet; this margin is defined in RFC 3261. The system default for the maximum UDP packet length is 1500.
  You can set the global SIP configuration's max-udp-length=X option for global use in your SIP configuration, or you can override it on a per-interface basis by configuring this option in a SIP interface configuration.
- udp-fallback—When a request needs to be sent out on the SIP interface for which you have configured this option, the Oracle USM first tries to send it over TCP. If the SIP endpoint does not support TCP, however, then the Oracle USM falls back to UDP and tries the request again.
  To enable SIP Transport Selection:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

```
ORACLE(configure)# session-router
```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option name **max-udp-length=X** (where X is the maximum UDP length you want to set), and then press Enter.

   ```
   ORACLE(sip-interface)# options +max-udp-length=900
   ```

   If you type options max-udp-length=X, you will overwrite any previously configured options. In order to append the new option to the sip-interface's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. **options**—Set the options parameter by typing **options**, a Space, the option name **udp-fallback**, and then press Enter.

   ```
   ORACLE(sip-interface)# options +udp-fallback
   ```

   If you type options udp-fallback, you will overwrite any previously configured options. In order to append the new option to the sip-interface's options list, you must prepend the new option with a plus sign as shown in the previous example.

6. Save and activate your configuration.

uaCSTA NAT Support

The Oracle USM offers User Agent Computer Supported Telecommunications Application (uaCSTA) support, which allows for the network address translation (NAT) of a key XML element in SIP INFO messages to use a phone's real contact URI.

Overview

Certain customers who use a uaCSTA for third party call control have encountered difficulties with the XML in their SIP messages used to support business applications. In these cases, the XML—specifically the <deviceID> XML tag—carries encoded IP addresses that need to be changed as they traverse the Oracle USM.

The SIP business application allows users to click-to-dial another party using e-mail application clients. The user's click triggers the application server to send a uaCSTA SIP INFO message through the system to the UA/phone. These SIP INFO messages contain XML with the user's Contact-URI. But the server is only aware of the Oracle USM's NAT'd Contact-URI and not the user's, so the XML in the SIP INFO is carrying incorrect information.

The XML element, then, needs to be NAT'd to the phone's real Contact-URI. This is especially important because of the broad use of SIP INFO messages, which instruct a phone to:

- Answer a call
- Hold a call
- Retrieve a call

All of these functions are available via a clickable interface on the e-mail application.

The Oracle USM performs the NAT to the <deviceID> XML tag only if it is configured to perform registration caching.

When the Oracle USM receives a SIP message from the core side and the request has:

- A Content-Type of application/csta+xml
- A Content-Length greater than 0

it parses the message's message body into an XML document. Should parsing fail, then the Oracle USM will forward the SIP INFO request without modification to the XML message body. Otherwise, the Oracle USM searches for the <deviceID> subelement within the XML document. If it finds the <deviceID> subelement, the Oracle USM searches through its registration cache for a registered Contact that matches the value of the <deviceID>. If it finds a match, the Oracle USM replaces the value of the <deviceID> with that of the corresponding registered Contact. If the value of the <deviceID> is a Contact that the Oracle USM generates for a registered UA, the corresponding contact from the look-up would be the Contact of the registered UA.

These functions performed, the Oracle USM then reformats the SIP INFO request with the modified XML message body before sending it to the next hop. If there is no match found, then the Oracle USM forwards the SIP INFO request without modifying the XML message body.

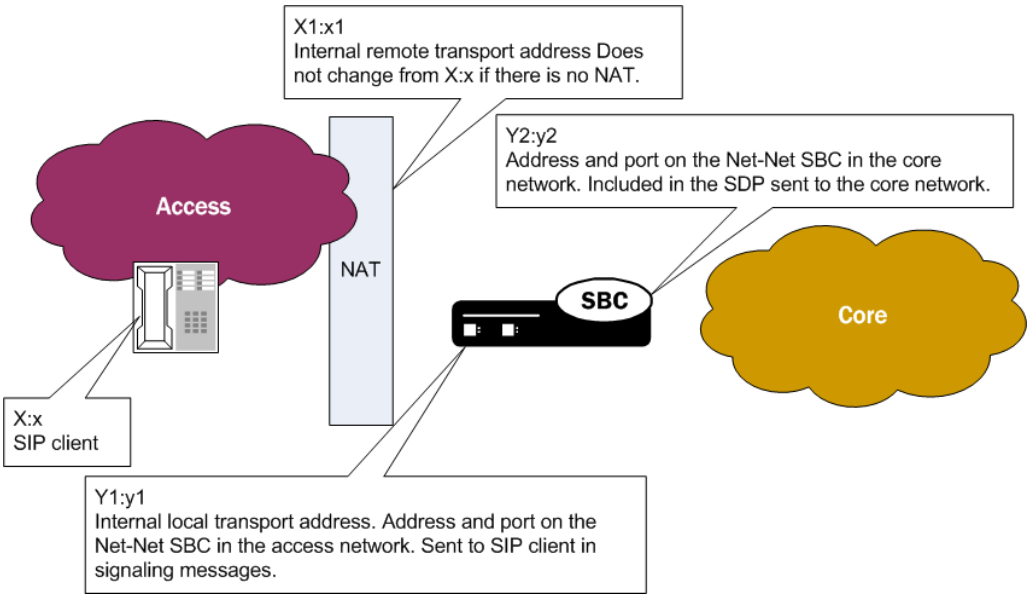
Other than ensuring your Oracle USM is configured to perform registration caching, you do not need take any further steps.

SIP Packet Cable Multi-Media

As a packet cable multi-media (PCMM) enhancement for SIP sessions key to next generation architectures, the Oracle USM can now include certain SDP attributes specifying media flow addresses in outgoing SIP messages. Previously, these address were hidden by the Oracle USM. Since SIP proxies and application servers in the core network, however, need to know these addresses to guarantee QoS for media flows in packet cable networks.

Certain options in the SIP interface configuration enable the Oracle USM to reveal address information on the core side.

When a SIP client in the access network sends and receives RTP media, the Oracle USM uses the SIP client's IP address and port (X:x) as its own internal remote transport address. The Oracle USM adds this information to outgoing SDP that it sends to the core side, and removes it from incoming SDP. If the SIP client sits behind a NAT, then the Oracle USM uses the IP address and port produced from the NAT (X1:x1) process for insertion and removal. The SIP client sends RTP to an IP address and port (Y1:y1) on the Oracle USM, referred to as the internal local transport address; this information is included in SDP (included in SIP messages) sent to the SIP client. Meanwhile, the Oracle USM also has an IP address and port (Y2:y2) in the core network. The far-end SIP UA sends RTP to this IP address and port, which are also included in SDP the Oracle USM sends to the core side.



To enforce QoS properly on the access side, the flow between the SIP client (or the SIP client's post-NAT IP address and port) and the internal local address must be revealed on the core side using SIP signaling messages.

Details

To enable this enhancement, you set three parameters in the SIP interface configuration:

- **sdp-internals**—Establishes that local and remote transport addresses need to be added.This option must be enabled on the access-side SIP interface, which is where the Oracle USM receives SDP.
- **sdp-local=<name>**—Sets a name for the internal local transport port address that the Oracle USM inserts into outgoing SDP. This option is configured on the core-side SIP interface. This address is removed from incoming SDP from the core side to prevent attributes from being sent back to the core in a hairpinned call.
- **sdp-remote=<name>**—Sets a name for the internal remote transport address that the Oracle USM inserts into outgoing SDP. This option is also configured on the core-side SIP interface. This address is also removed from incoming SDP from the core side to prevent attributes from being sent back to the core in a hairpinned call.

Further, the Oracle USM determines whether or not to insert the SDP attributes based on a call's ingress and egress signaling realms:

Address Information	Calling-Side SDP	Called-Side SDP
Internal local transport address	Added to SDP when: The ingress signaling realm's SIP interface has the sdp-internals option configured The egress signaling realm's SIP interface has a defined sdp-local option	Added to SDP when: The egress signaling realm's SIP interface has the sdp-internals option configured The ingress signaling realm's SIP interface has a defined sdp-local option

Address Information	Calling-Side SDP	Called-Side SDP
Internal remote transport address	Added to SDP when: The ingress signaling realm's SIP interface has the sdp-internals option configured The egress signaling realm's SIP interface has a defined sdp-remote option	Added to SDP when: The egress signaling realm's SIP interface has the sdp-internals option configured The ingress signaling realm's SIP interface has a defined sdp-remote option

Core-Side SDP Insertion Configuration

In a typical configuration intended to send SDP to the core side with the inserted attributes, the access SIP interfaces have the **sdp-internals** option enabled, and the core SIP interfaces have the **sdp-local** and **sdp-remote** values configured.

To set the access SIP interface for SDP insertion on the core side:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option name **sdp-internals** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +sdp-internals
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

   To set the local and remote transport addresses for a core SIP interface:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

7. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

8. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# sip-config
ORACLE(sip-interface)#
```

9. **options**—Set the options parameter by typing **options**, a Space, the option name preceded by a plus sign (+) (**sdp-local=<name>**, where the name is attribute name for the SDP), and then press Enter. Follow the same steps to add the **sdp-remote** option.

   ```
   ORACLE(sip-interface)# options +sdp-local=Local_Turn
   ORACLE(sip-interface)# options +sdp-remote=PCMM_USERADD
   ```

   If you type either of these options without the plus (+) sign, you will remove any previously configured options. In order to append the new option to the options list, you must prepend the new option with a plus sign as shown in the example above.

10. Save and activate your configuration.

SIP Method-Transaction Statistic Enhancements

In prior releases, the Oracle USM tracks SIP session agents, SIP interfaces and SIP realms on a global level. Only counters that are related to session rates and constraints are displayed.

You can now enable your Oracle USM to track transaction messages for specific SIP session agents, SIP realms, and SIP interfaces.

The following SIP methods are tracked for Recent, Total, and Period Max values:

- INVITE | ACK | BYE | REGISTER | CANCEL | PRACK | OPTIONS | INFO | SUBSCRIBE | NOTIFY | REFER | UPDATE | MESSAGE | PUBLISH | other (unknown)

With this new tracking enhancement, the **show sipd** command has been updated with a new method argument which allows you to query statistics for a particular method for a given SIP agent, SIP interface, or SIP realm.

SIP Method Tracking Enhancements Configuration

To enable or disable the expanded SIP Method statistics tracking:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ```

4. **extra-method-stats**—Enable this parameter if you want to use the expanded SIP Method tracking feature. The default is **disabled**. The valid values are:

   - enabled | disabled

5. Save and activate your configuration.

National Security and Emergency Preparedness for SIP

The Oracle USM supports Emergency Telecommunications Service (ETS), which gives priority treatment of National Security and Emergency Preparedness (NSEP) communications for IP network infrastructures. ETS can increase the likelihood that calls, sessions, and other communications will be successfully completed when they are initiated by government-authorized users over the public network infrastructure. Legacy circuit-switched services such as Government Emergency Telecommunications Service (GETS) and Wireless Priority Service (WPS) also fall under the ETS rubric, and are now also supported on the Oracle USM.

To provide this support, you can enable the Oracle USM to act on SIP calls that contain an ETS dial number (DN) and/or the SIP Resource-Priority header that carries ETS resource values.

The Oracle USM identifies ETS calls by using the system's pre-existing network management controls (NMC) functionality. With NMC and Resource-Priority header (RPH) support enabled on your system, the Oracle USM detects ETS calls and provides the appropriate treatment for them.

The Oracle USM supports this feature by treating ETS calls based on the r-value parameter in the Resource-Priority header. The r-value is a key piece of information because it defines the resource priority that the call originator requests. The r-value parameter provides namespaces and priorities that the Oracle USM can manipulate in outgoing traffic.

In addition to a new RPH profile configuration containing information about how to treat RPHs, new parameters in the global SIP configuration and NMC configuration have been added. The RPH profile is applied to an NMC rule, where they determine r-values, a media policy to use, and what type of call treatment to apply. Also applies to an NMC rule, the new RPH policy configuration provides information about which r-values to insert and which to override.

Licensing

To enable NSEP for SIP on your Oracle USM, you must obtain and install a new license. If properly installed on your system, it appears as NSEP RPH in the display issued when you use the ACLI **show** command in the license configuration.

For information about how to obtain an NSEP RPH license, contact your Oracle sales representative.

Matching by NMC and by RPH

When a Oracle USM has been enabled to act on RPH, it checks incoming requests for RPH, tries to parse that RPH, and then rejects requests in the circumstances listed below. For all of these rejections, the Oracle USM logs the error at the TRACE level.

- Request with multiple instances of the same namespace in the RPH—The Oracle USM sends out a 400 Bad Request response with the Invalid RPH - Namespace repeated header showing that there are multiple instances of the same namespace in the RPH.
- Request with invalid resource priority for a namespace—The Oracle USM sends out a 400 Bad Request response with the Invalid RPH - Invalid rvalue: x showing that there is an invalid resource value (where x is the invalid value).

- Request with WPS namespace, but without ETS namespace—The Oracle USM sends out a 400 Bad Request response with the Invalid RPH - No ETS value header showing that there is no ETS namespace.

If the Oracle USM successfully parses the RPH, it identifies the ETS call by checking the Request-URI of the incoming request against destination identifiers that you configure in the NMC rules. If there is a match between the request's ETS DN and the destination value identifier in the NMC rules, the Oracle USM tags the call; note that NMC rules need to be configured with the **rph-feature** parameter set to enabled to identify an ETS call properly. If there is no match to an NMC rule, then the Oracle USM performs matching based on RPH by comparing resource values (r-values) in the RPH with values you set in the RPH profile configuration.

For an ETS call that matches by ETS DN and NMC rule, the system checks the NMC rule to determine if it has an RPH profile (with r-values) assigned to it. If so, the Oracle USM continues by comparing the RPH profile's r-values against those in the request's RPH. In cases where the RPH does not contain a recognized value r-value, the Oracle USM:

- Processes the call as it normally would (as a non-ETS call) without changing the RPH if the resource-priority option tag is not present in the Required header (for an INVITE only and not any other requests or response from which RPH would be deleted)
- Rejects the Request when the Require header has the resource-priority header; or, inserts an Accept-Resource-Priority header (ARPH) in the response if the **insert-arp-header** parameter option is enabled

However, the call goes through the Oracle USM as an ETS call when it is matched by ETS DN and the applicable NMC does not have an RPH profile assigned. According to the settings in the NMC rule, the Oracle USM either diverts or rejects such a call. And when the call matches by RPH rather than ETS DN, the Oracle USM applies the configured RPH profile from the relevant NMC rule.

It can be the case that non-ETS calls have RPH in their requests. Here, the Oracle USM call treatment is performed according to the settings in the matching RPH profile when there is no matching NMC rule. When you configure treatment as "reject," then the Oracle USM rejects the call with a 417 Unknown-Resource Priority status code. When you set the treatment to either "accept" or priority, the Oracle USM allows the call to proceed as a non-ETS call or as a priority call.

The ETS r-value can appear in ACK, BYE, INFO, PRACK, REFER and UPDATE requests. In cases when it does and the session with which the request is associated is a non-ETS call, the Oracle USM removes the RPH from the request before forwarding it and logs a TRACE-level error. The Oracle USM also removes RPH from responses before forwarding them and logs a TRACE-level error when responses contain RPH headers with ETS values for non-ETS sessions.

Call Treatment

This section describes how ETS calls are treated as they traverse the Oracle USM.

Call Treatment	Description
Routing	ETS calls are routed the same way as any other calls are, except when the applicable NMC rule's treatment type is divert, and rule defines the next hop. This route takes precedence over other normal routes.

Call Treatment	Description
Local NMC	ETS calls are exempt from the local NMC, including: session agent constraints, bandwidth constraints (e.g., per-realm bandwidth), per-user CAC, and CPU constraints. However, the call is subject to the ETS congestions control threshold. Licensing session constraints apply.
ETS Call Congestion Control	ETS calls are subject to congestion control constraints that you configure specifically for this type of traffic. In the global SIP configuration, you set up one option that defines a load limit (greater than that set for normal calls).
ETS CAC	Although the Oracle USM uses the call rate control value in the applicable NMC rule, you can also enforce call rate on a per-user basis for ETS calls.

When the Oracle USM receives a SIP INVITE with an RPH matching an NMC with an ETS DN, but whose r-values do not match the NMC's rph-profile, the Oracle USM behaves as follows:

- If the INVITE does not have the resource-priority option tag and:
  - If the matching NMS is set to PRIORITY, the call will be treated as an NSEP call. If there is an rph-profile matching the r-value (not necessarily the one in the NMC), the Oracle USM uses the media-policy from that rph-profile for the call. The rph-policy from the NMC (if present) also applies to the call.
  - If the matching NMC is not set to PRIORITY, the Oracle USM will treat the call as a normal one.

If the INVITE contains the resource-priority option tag, the Oracle USM will reject the call with the 417 Unknown Resource-Priority message.

Generating Egress RPH

For each ETS call, the Oracle USM generates RPH for the outgoing request. It forms this RPH according to the information in the NMC rule. The outgoing request types are INVITE, ACL, BYE, CANCEL, INFO, PRACK, REFER, and UPDATE.

Request RPH Status	Generated Egress RPH
Incoming request without RPH (matched by ETS DN)	Outgoing RPH value becomes the r-value set in the insert-r-value parameter in the RPH policy applied to the NMC rule.
Incoming request without RPH (matched by ETS DN)	If the insert-r-value parameter is empty in the RPH policy applied to the NMC rule or there is no RPH policy applied to the NMC rule, then the egress RPH will also not have RPH.
Incoming request has RPH	Egress RPH is the same as the ingress if the NMC rule has an RPH policy applied but the override-r-value for the policy is empty or if there is not RPH policy applied to the NMC rule. If the override-r-value for the policy is set, then the egress RPH is set to that value.

For example, given an incoming request with the resource priority ets.0, dsn.flash and an RPH policy with an override value of wps.1,ets.1, the egress request would be sent with a resource-priority of wps.1,ets.1,dsn.flash.

The Oracle USM also includes RPH in the following series of responses, even when the downstream SIP entity does not respond with an RPH: 1xx, 2xx, 3xx, 4xx, 5xx, and 6xx. The 401 Unauthorized response is an exception.

Media Treatment

If the RPH profile set in an NMC names a media policy, then the Oracle USM implements it for the ETS call. This media policy overrides any media policy set in the realm configuration.

The possible Differentiated Services Code Point (DSCP) values for an ETS call are:

- Audio—Applied to the respective media for an ETS call
- Video—Applied to the respective media for an ETS call
- SIP—Applied to the ETS calls' SIP signaling messages, only for the egress call leg for the ETS session

RPH Configuration

This section shows you how to configure RPH profiles and policies that enable the Oracle USM to act on SIP calls that have an ETS DN and/or an RPH carrying ETS resources values. There are also settings for the global SIP configuration and for the NMC rule configuration that support this feature.

In addition, note that:

- You must set a media policy for the RPH profile to use. Check your system configuration and note the name of the media policy that best suits your needs.
- Valid values for the parameters that take r-values are wps.x and ets.x, where x is 0 through 4.

Remember to save and activate your configuration after you have completed the processes detailed in this section.

Setting Up and Applying RPH Policy

The RPH policy is a configuration on the Oracle USM that you apply to NMC rules. It designates the following for ETS/WPS namespaces:

- An override resource value—Resource value used to override the incoming RPH's resource value
- An insert resource value—Resource value inserted when the Oracle USM does not recognize the RPH, the incoming request has no RPH, or the call is H.323 and matches an NMC rule based on the ETS DN

Note that RPH policies do not apply for DSN, DRSN, Q.735, or any other type of namespace; these remain untouched in outgoing requests.

To configure an RPH policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **rph-policy** and press Enter. From here, you can configure the individual parameters for the RPH policy.

```
ORACLE(session-router)# rph-policy
ORACLE(rph-policy)#
```

4. **name**—Enter the name that uniquely identifies this RPH policy. This is the value you use to apply the policy in the NMC rules configuration. There is no default for this parameter, and you are required to set it.
5. **override-r-value**—Enter the value that the system uses to override r-values in the original RPH.

   ```
   ORACLE(rph-policy)# override-r-value ets.1
   ```

6. **insert-r-value**—Enter the value that the Oracle USM inserts into the RPH.

   ```
   ORACLE(rph-policy)# insert-r-value wps.1
   ```

7. **rph-policy**—Enter the name of the RPH policy that you want to apply for this NMC rule. This parameter is empty by default; if you do not set an RPH policy, none will be applied.

Setting Up and Applying RPH Profile

The RPH profile contains information about how the system should act on the namespace(s) present in a Resource-Priority header (if any). The list of resource values in this configuration calls out the resource values (or r-values) recognizable to the Oracle USM; the ETS and WPS namespaces are supported.

You also set a media policy for the RPH profile to use; it defines the Differentiated Services Code Point (DSCP) that the Oracle USM uses for media or signaling packets belonging to the egress call leg for the ETS session.

The call treatment parameter tells the Oracle USM what to do with a non-ETS call that has RPH in its request; the call can be allowed, rejected, or treated as a priority call.

To configure an RPH profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **rph-profile** and press Enter. From here, you can configure the individual parameters for the RPH policy.

   ```
   ORACLE(session-router)# rph-profile
   ACMEORACLEPACKET(rph-profile)#
   ```

4. **name**—Enter the name that uniquely identifies this RPH profile. This is the value you use to apply the profile in the NMC rules configuration. There is no default for this parameter, and you are required to set it.
5. **r-values**—Enter one or more r-values that the Oracle USM is to recognize for matching purposes. When you enter more than one value in the list, you type the name of the parameter followed by a Space, open quotation mark, the values for the list separated by spaces, a closed quotation mark. Then press Enter.

   You must enter them in the order reflected below (a WPS and then an ETS value). A WPS call always has to have an ETS namespace.

   ```
   ORACLE(rph-profile)# r-values "wps.0 ets.2"
   ```

6. **media-policy**—Enter the name of a media policy configuration that you want applied for this RPH profile. The Oracle USM implements this media policy for the ETS call, and this media policy overrides any media policy set in the realm configuration.
7. **call-treatment**—Enter the call treatment method for a non-ETS call that contains RPH matching it to this profile. The default is accept. The valid values are:
   - **accept**—The call proceeds as it normally would
   - **reject**—The Oracle USM rejects the call with the 417 Unknown-Resource Priority status code
   - **priority**—The Oracle USM treats the call as a priority call

Enabling NSEP for an NMC Rule

In addition to the RPH policy and RPH profile you can set for an NMC rule, you also need to set the state of this feature for the NMC rule.

To enable NSEP for an NMC rule:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ORACLE(net-management-control)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **rph-feature**—Enable this parameter if you want to turn the NSEP feature on for this NMC rule. The default is **disabled**. The valid values are:
   - enabled | disabled

Global SIP Configuration Settings Enabling NSEP

For the global SIP configuration, you can turn the NSEP feature on, and you can also set parameters that support call admission and congestion control.

In addition, you can enable the insertion of the ARPH header in a response when the resource-priority tag is present in the Require header and the Oracle USM rejects the request with a 417 Unknown Resource-Priority response. The ARPH value is the list of r-values you set in the RPH profile.

To enable NSEP for the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **rph-feature**—Enable this parameter if you want to turn the NSEP feature on for the global SIP configuration. The default is **disabled**. The valid values are:

   - enabled | disabled

Global SIP Configuration Settings Enabling CAC and Congestion Control

To set call admission and congestion control parameters for NSEP:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **nsep-user-sessions-rate**—Enter the maximum INVITEs per second to admit for ETS calls on a per-user basis. To enable NSEP call admission control (CAC), you must change the parameter value from 0; if you leave this parameter set to 0, then it is the same as disabling CAC for ETS calls. The default is **50**. The valid range is:

   - Minimum—0
   - Maximum—999999999

5. **options**—To enable congestion control for ETS calls, you configure an option that sets the CPU threshold. If this threshold is exceeded, the Oracle USMrejects new ETS calls with the 503 Service Unavailable response. The value you set here should be larger than the load limit value for normal calls; ETS calls are allowed even when the load limit threshold for normal calls is exceeded.

   The threshold value can be between 0 and 100. Using a value of 0 or 100 for this parameter disables ETS call congestion control.

   Set the options parameter by typing **options**, a Space, the option name **nsep-load-limit** with a plus sign in front of it, then the equal sign and the ETS call threshold you want to set. Then press Enter.

   ```
   ACMEPACKET(sip-config)# options +nsep-load-limit=50
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

Global SIP Configuration Settings Enabling ARPH Insertion

To enable ARPH insertion in responses:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   If you are adding support for this feature to a pre-existing configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **options**—To enable ARPH insertion in responses type **options**, a Space, the option name **insert-arp-header** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-config)# options +insert-arp-header
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

Setting Up NSEP for Session Agents

In earlier releases, the Oracle USM supports NSEP-related CAC for users and for NMC. You can now configure a sessions-per-second rate for session agents. Set in the global SIP configuration, this rate applies to all SIP session agents. When session exceed the limit, the Oracle USM rejects them with a 503 Service Unavailable message.

To configure NSEP limits for SIP session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **nsep-sa-sessions-rate**—Enter maximum acceptable number of SIP INVITES (NSEP sessions) per second to allow for SIP session agents. This parameter defaults to 0, meaning there is no limit.

5. Save and activate your configuration.

E-CSCF Emergency Setting Precedence for NMC

When the Oracle USM acts as an E-CSCF, it can route emergency or priority calls (i.e., 112, 911, 999 calls) to the corresponding ECS/PSAP based on the calling party's information. However, for registered users, this ability mixes with the Oracle USM's NMC function so that the Service Routes takes precedence over the NMC. So rather than routing the emergency call to the ECS/PSAP, the call ends up at the S-CSCF of the Service Route.

For non-registered users where there no Service Route exists, this is not an issue.

Adding the **apply-local-policy** value to the options parameter in the network management controls configuration allows the NMC to take precedence over cached Service Route when a session matches an NMC rule. Instead, it locates a matching local policy.

Without the NMC configured to take precedence, the Oracle USM does not function optimally as an E-CSCF for registered users.

Consider the following scenario in which a UE registers as a P-CSCF to the S-CSCF via the Oracle USM. In this case, the registrar returns a Service Route header in a 200 OK response to the REGISTER message to the Oracle USM. Or, if an implicit service route is enabled, the Oracle USM generates a Service Route that it saves in the register cache before forwarding the 200 OK on to the UA. Then when a new INVITE comes from the UA, the Oracle USM checks its register cache and uses the Service Route as the next hop—without taking local policy into consideration.

Using the **apply-local-policy** value requires local policy's consideration in deciding the next hop. If you configure this option and the Oracle USM receives a new INVITE, it will decide whether the session is priority, registered, or with Service Route. Then it will determine if the call is E-CSCF. If so and the first matching local policy has E-CSCF enabled, the priority local policy is applied over the Service Route. The Oracle USM stays with the Service Route if not.

E-CSCF Emergency Configuration

This section shows you how to configure a network management control rule to take precedence over a Service Route.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ORACLE(net-management-control)#
   ```

4. **options—Follow your entry with this value:**

   - +disable-temp-file

     ```
     ORACLE(net-management-control)# options +apply-local-policy
     ```

     This value allows a network management control rule to take precedence over a service route for an emergency or priority call.

5. Type **done** and continue.

SIP TCP Connection Reuse

You can configure your Oracle USM to reuse TCP connections created by SIP peering devices for outgoing SIP in-dialog and out-of-dialog request transactions.

The SIP draft draft-ietf-sip-connect-reuse-07.txt describes a way for SIP UAs to reuse connections created by a remote endpoint for outgoing requests for TLS. The Oracle USM does not support the model connection reuse signalled by a parameter; rather, it is provisioned on a per-session-agent basis.

You enable SIP TCP connection reuse on a per-session-agent basis. The Oracle USM checks incoming TCP connection request to determine if they are from session agent that has this feature turned on. When it is, the Oracle USM adds the connection's source address to its list of alias connections. This is a list of connections that the Oracle USM can use for outgoing requests rather than creating its own connection (as it does when this feature is not enabled). So if a preferred connection fails, the Oracle USM can refer to this list and use the alias connection.

The presence of an alias parameter in the Via header is just one mechanism that will call the Oracle USM to use the inbound TCP/TLS connection for outbound requests. The Oracle USM will automatically add an alias for the inbound connections in the following circumstances:

- The other end of the connection is behind a NAT. When the Oracle USM sees that the Via sent-by does not match the source address of the connection, it will automatically reuse the connection to deliver requests to the UA.
- The Contact address of a REGISTER request received on a TCP connection matches the source address and port. This is because the contact adress is the ephemeral port the UA used to form the connection to the Oracle USM and, therefore, will not be listening on that port for inbound connections.
- The presence of reuse-connections in the options field of the sip-interface will cause the Oracle USM to reuse all inbound TCP connections for sending rquests to the connected UA.

SIP TCP Connection Reuse Configuration

This section describes how to enable SIP TCP connection reuse for a session agent. Currently there are two options for the new **reuse-connections** parameter: none (which turns the feature off) and tcp (which enables the feature for TCP connections). You also set the re-connection interval.

To enable SIP TCP connection reuse for a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

4. **reuse-connections**—Enable or disable SIP TCP connection reuse. The default is **none**. This value disables the feature. The valid values are:
   - tcp | none
5. **tcp-reconn-interval**—Enter the amount of time in seconds before retrying a TCP connection. The default for this parameter is **0**. The valid range is:
   - Minimum—0, 2
   - Maximum—300
6. Save and activate your configuration.

SIP TCP Keepalive

The Oracle USM supports a special TCP keepalive mechanism for SIP. By enabling this feature either for a session agent or for a SIP interface, you allow the Oracle USM to use standard keepalive probes to determine whether or not connectivity with a remote peer has been lost.

This feature adds to the Oracle USM's pre-existing TCP keepalive functionality that you can enable in the network parameters configuration. Using existing functionality, you can customize keepalive timing by:

- Specifying the number of unacknowledged packets the Oracle USM sends to the remote peer before it terminates the TCP connection.
- Specifying the number of seconds of idle time before TCP keepalive messages are sent to the remote peer.

You can now set three modes for TCP keepalive for session agents and SIP interfaces:

- none—(Default) Keepalives are not enabled for use with the session agent/SIP interface; when you select this setting for a session agent, it will use the setting for this feature from the SIP interface.
- enabled—Keepalives are enabled for the session agent/SIP interface.
- disabled—Keepalives are disabled for the session agent/SIP interface.

Note that the setting for this feature for a session agent takes precedence over that for a SIP interface. In addition, the session agent offers you a way to set the re-connection interval.

SIP TCP Keepalive Configuration for Session Agents

To enable SIP TCP keepalive for session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the session agent that you want to edit.

4. **tcp-keepalive**—Enable or disable standard keepalive probes to determine whether or not connectivity with a remote peer is lost. The default value is **none**. The valid values are:
   - none | enabled | disabled

     ```
     ACMEPACKET(session-agent)# tcp-keepalive enabled
     ```
5. Save and activate your configuration.

SIP TCP Keepalive Configuration for SIP Interfaces

To enable SIP TCP keepalive for SIP interfaces:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```
2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```
3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing SIP interface, then you must select (using the ACLI **select** command) the SIP interface that you want to edit.
4. **tcp-keepalive**—Enable or disable SIP TCP keepalive. The default value is **none**. The valid values are:
   - none | enabled | disabled

     ```
     ORACLE(session-agent)# tcp-keepalive enabled
     ```
5. Save and activate your configuration.

SIP Enforcement Profile and Allowed Methods

For this feature, you use a configuration called an enforcement profile that allows you to configure sets of SIP methods that you want applied to: the global SIP configuration, a SIP interface, a realm, or a SIP session agent. The enforcement profile is a named list of allowed methods that you configure and then reference from the configuration where you want those methods applied.

SIP Enforcement Profile Configuration

To use the enforcement profile, you need configure it with a name and the list of SIP methods you want to designate as allowed. Then you need to configure the global SIP configuration, a SIP interface, a realm, or SIP session agent to use the set.

Setting Up and Enforcement Profile

To set up an enforcement profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **enforcement-profile** and press Enter.

   ```
   ORACLE(session-router)# enforcement-profile
   ORACLE(enforcement-profile)#
   ```

4. **name**—Enter the name for the enforcement profile. This parameter has no default, but you must note it so that you can apply this set of allowed SIP headers in: the global SIP configuration, a SIP interface, a realm, or SIP session agent.

   ```
   ORACLE(enforcement-profile)# name EnfProfile1
   ```

5. **allowed-methods**—Enter a list of SIP methods that you want to allow for this set. The default value is **none**. Valid values are:

   - INVITE | REGISTER | PRACK | OPTIONS | INFO | SUBSCRIBE | NOTIFY | REFER | UPDATE | MESSAGE | PUBLISH

     To enter multiple methods for the list, type the parameter name followed by a space, then the names of all methods you want to include each separated by a only a comma and in capital letters.

     ```
     ORACLE(enforcement-profile)# allowed-methods INVITE,REGISTER,PRACK
     ```

6. Save and activate your configuration.

Applying an Enforcement Profile

You can apply an enforcement profile to: the global SIP configuration, a SIP interface, a realm, or SIP session agent. This section shows you how to do all four. Remember that if you are adding this functionality to a pre-existing configuration, you need to select the configuration you want to edit.

To apply an enforcement profile to the global SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to the global SIP configuration.
5. Save and activate your configuration.

   To apply an enforcement profile to a SIP interface:
6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```
7. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```
8. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```
9. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this SIP interface.
10. Save and activate your configuration.

    To apply an enforcement profile to a SIP session agent:
11. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```
12. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```
13. Type **session-agent** and press Enter.

    ```
    ORACLE(session-router)# session-agent
    ORACLE(session-agent)#
    ```
14. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this session agent.
15. Save and activate your configuration.

    To apply an enforcement profile to a realm:
16. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```
17. Type **media-manager** and press Enter.

    ```
    ORACLE(configure)# media-manager
    ORACLE(media-manager)#
    ```
18. Type **realm-config** and press Enter.

    ```
    ORACLE(media-manager)# realm-config
    ORACLE(realm-config)#
    ```
19. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm.
20. Save and activate your configuration.

Local Policy Session Agent Matching for SIP

When you enable the local policy session agent matching option in your global SIP configuration, you change the way local policies match session agents. Normally, the Oracle USM looks up and stores matched session agents configured as next hops so it does not need to perform the lookup while processing requests. In this type of matching, the Oracle USM does take the realm set in the local policy attributes into consideration. When the Oracle USM performs its regular matching method and you have enabled overlapping IP addresses for session agents, the Oracle USM might match session agents to different realms than the ones you intended when creating your configuration.

Local policy session agent matching provides a way to match session agents differently, taking realms and nested realms into consideration during the matching process. This difference is key to deployments with multiple peering partners that use the overlapping IP address feature, and have multiple local policies routing to the same IP address in different realms where some target next hops require session constraints but others do not. In the cases where no session constraints are required, session agents are not needed. But session agents still match the local policy, applying their constraints, because they match the next hop IP address.

In addition to modifying this behavior, this feature also affects the use of realms and nested realms. It triggers the use not only of realms, but of all the realms nested however deeply—thereby improving matching efficiency.

You can set the local policy session agent matching option with values that define how the Oracle USM performs session agent matching:

- **any**—The Oracle USM looks up and stores matched session agents configured as next hops so it does not need to perform the lookup while processing requests, without regard to realms.
  This behavior is the default when the SIP configuration does not have the local policy session agent matching option set.
- **realm**—The Oracle USMselects session agents in the realm that the local policy attribute indicates; this provides an exact match, rather than not taking the realm into consideration during session agent selection.
  For example, the session agent is a match if the session agent **realm-id** and the local policy attribute **realm** parameters are an exact match.
- **sub-realm**—Session agents in the same realm or the same realm lineage—where session agents and realms are related to one another via realm parent-child relationships no matter the depth of realm nesting configured
  For example, the session agent is a match if the local policy attribute **realm** is a sub-realm of the realm specified in the session agent **realm-id** parameter.
- **interface**—Session agents in the same realm or same realm lineage via the realm set in the local policy attribute, and whose realm uses the same signaling interface as the realm set in the local policy attribute
  For example, the session agent is a match if the session agent **realm-id** is a sub-realm of the local policy attribute **realm**, and both referenced realms use the same SIP signaling interface.
- **network**—Session agents whose realm is in the realm lineage for the same realm set in the local policy attributes, and whose realm is associated with the same network interface as the realm set in the local policy attributes
  For example, the session agent is a match if the session agent **realm-id** is a sub-realm of the local policy attribute **realm**, and realm reference by both use the same network interface.

If it cannot find a match, the Oracle USM will use the IP address as the next hop. Further, requests matching local policy attributes will not be associated with session agents, and so their constraints will not be applied.

The Oracle USM stores session agent information that it looks up when performing local policing session agent matching. To perform the lookup, it uses the session agent hostname as a key. When the hostname is an FQDN and there is a configured IP address in the **ip-address** parameter, the Oracle USM uses the ip-address value as a secondary key. Given this implementation, the following are true when selecting session agents:

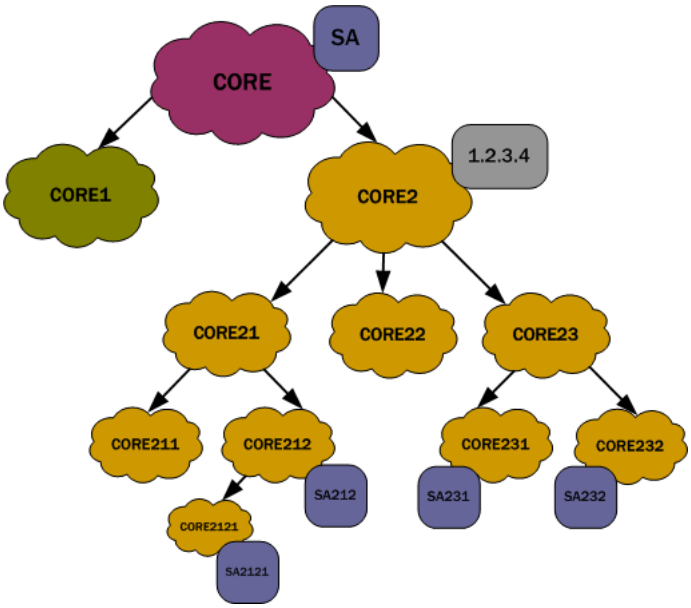
- If multiple session agents share the same IP address, the one with an IP address in the hostname parameter takes precedence.
- If all session agents with the same IP address have an FQDN as their hostname, the one whose name is alphabetically lower will take precedence, where alphabetically lower means earlier in the alphabet (closer to A than to Z).
- For non-global session agents (whose realms are configured but not wildcarded) with an IP address, the Oracle USM uses a key that is a combination of the IP address and the realm in the form <address>:<realm>.
- For a session agent whose realm has a parent realm, the Oracle USM uses a combination of the IP address, realm, and realm-path (or lineage for the realm) in the form <address>:<realm-path>. For example, the realm path for a realm core3 with a parent core2, which in turn has a parent core would be core:core2:core3.

When it looks up a session agent with a realm, the Oracle USM first searches for an exact match for the IP address and realm combination. If this fails, it performs a second search if the desired realm has parents or children. The Oracle USM locates an entry in its repository of session agent information that is greater than or equal to the IP address with the base realm, which is the ancestor of the desired realm without a parent. Having gathered this set of candidates, the Oracle USM narrows down the search for a match by comparing sub-realms and determines there is a match if either:

- The desired realm path is a sub-string of the entry's realm path, or
- The entry's realm path is a substring of the desired realm path (i.e., the desired realm is a sub-realm of the entry's realm)

Then the Oracle USM orders the candidates by depth of the entry's realm-path, or number of levels from the base realm relative to the depth of the desired realm. By searching the ordered set until the entry's realm depth equals the desired realm's depth, the Oracle USM determines a parent candidate, all subsequent entries being sub-realms of the desired realm. The Oracle USM only considers entries at the first level deeper than the desired realm. If at this point there is only one entry, the Oracle USM deems it a match. Otherwise, it selects the parent candidate as the matching entry. In the event the search does not yield a matching realm, the Oracle USM uses the global session agent for the IP address, if there is one.

The following diagram shows the realm tree, where the clouds are realms and squares are session agents, representing a group of session agents sharing the IP address 1.2.3.4. The Oracle USM searches for the session agents lower in the tree along the session agent realm-path and the desired realm.



For the diagram above, the following shows how the hostname would look for this group of session agents.

Key	Session Agent (hostname[realm])
1.2.3.4 (This session agent owns the primary key for the IP address because its hostname is the IP address.)	1.2.3.4[CORE2]
1.2.3.4:CORE (IP+realm key entry)	SA[CORE]
1.2.3.4:CORE (IP+realm key entry)	1.2.3.4[CORE2]
1.2.3.4:CORE212 (IP+realm key entry)	SA212[CORE212]
1.2.3.4:CORE2121 (IP+realm key entry)	SA2121[CORE2121]
1.2.3.4:CORE231 (IP+realm key entry)	SA231[CORE231]
1.2.3.4:CORE232 (IP+realm key entry)	SA232[CORE232]
1.2.3.4:CORE: (IP+realm-path key entry)	SA[CORE]
1.2.3.4:CORE:CORE2: (IP+realm-path key entry)	1.2.3.4[CORE2]
1.2.3.4:CORE2:CORE21:CORE212 (IP+realm-path key entry)	SA212[CORE212]
1.2.3.4:CORE2:CORE21:CORE212:CORE2121 (IP+realm-path key entry)	SA2121[CORE2121]
1.2.3.4:CORE2:CORE23:CORE231 (IP+realm-path key entry)	SA231[CORE231]
1.2.3.4:CORE2:CORE23:CORE232 (IP+realm-path key entry)	SA232[CORE232]

For each realm in the table above, the search results for each realm would look like this:

IP Address	Realm	Session Agent (hostname[realm])
1.2.3.4	CORE	SA[CORE]
1.2.3.4	CORE2	1.2.3.4[CORE2]
1.2.3.4	CORE21	SA212[CORE212[
1.2.3.4	CORE211	1.2.3.4[CORE2]
1.2.3.4	CORE212	SA212[CORE212]
1.2.3.4	CORE2121	SA2121[CORE2121]
1.2.3.4	CORE22	1.2.3.4[CORE2]
1.2.3.4	CORE23	1.2.3.4[CORE2]
1.2.3.4	CORE231	SA231[CORE231]
1.2.3.4	CORE232	SA232[CORE232]

Local Policy Session Agent Matching Configuration

When you enable local policy session agent matching, remember that you can choose from five different ways to use the feature: **all**, **realm**, **sub-realm**, **interface**, and **network**.

This example shows you how to use the **realm** selection.

To enable local policy session agent matching using the realm method:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-config** and press Enter.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name **lp-sa-match=X (where X is the local policy session agent matching method you want to use)** with a plus sign in front of it. Then press Enter.

   Remember that if you do not specify a method, the system uses the **all** method.

   ```
   ORACLE(sip-config)# options +lp-sa-match=realm
   ```

   If you type **options** and then the option value for either of these entries without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

- Unordered—Meaning that the endpoint can deliver data within regard for their stream sequence number

You set this preference in the network parameters configuration.

About Wildcarding

The Oracle USM supports wildcarding the event type in the **subscribe-event** configuration. To wildcard the value, you enter an asterisk (*) for the **event-type** parameter instead of typing in the name of an actual event type.

When you wildcard this value, the Oracle USM applies the subscription limitations you set across all event types. Or, if you have entered multiple subscribe-event configurations, the Oracle USM applies the wildcard limits across the event types for which you have not set limits.

Consider the following example of a configured enforcement profile with a wildcarded **subscribe-event** configuration:

```
enforcement-profile
        name                            rulefour
        allowed-methods
        sdp-address-check               disabled
        subscribe-event
                 event-type                     *
                 max-subscriptions              1
        subscribe-event
                 event-type                     xyz
                 max-subscriptions              0
        last-modified-by                admin@console
        last-modified-date              2008-11-11 12:49:27
```

In this example, the enforcement profile allows all subscriptions that are event type xyz for a user. But it allows only one maximum for every other subscription event type.

Monitoring

You can display the number of subscription dialogs per SUBSCRIBE event type using the ACLI **show registration sipd subscriptions-by-user** command. You can display this information per event type, or you can show data for all event types by wildcarding the event type argument.

Enforcement Profile Configuration with subscribe-event

This section shows you how to configure an enforcement profile with a **subscribe-event** configuration. Remember that you can set up multiple **subscribe-event** configurations to correspond with the event types you want to control. It also shows you how to apply these limitations to a realm.

Setting Up Subscribe Dialog Limits

Setting up subscribe dialog limits means setting up an enforcement profile. For the sole purpose of setting up the subscription event limits, you only need to configure the name parameters and then as many **subscribe-event** configurations as you require. The enforcement profile has other uses, such as SIP SDP address correlation, so only configure the parameters you need.

To configure subscribe dialog limits:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **enforcement-profile** and press Enter.

```
ORACLE(session-router)# enforcement-profile
ORACLE(enforcement profile)#
```

4. **name**—Enter a name for this enforcement profile. You will use this name later when you apply the enforcement profile to a realm; it is the value you enter into the **enforcement-profile** parameter in the realm configuration.
5. Still in the enforcement profile configuration, type **subscribe-event** and press Enter.

```
ORACLE(enforcement profile)# subscribe-event
ORACLE(subscribe-event)#
```

6. **event-type**—Enter the SIP subscription event type for which you want to set up limits. You can also wildcard this value (meaning that this limit is applied to all event types except the others specifically configured in this enforcement profile). To use the wildcard, enter an asterisk (*) for the parameter value.

   By default, this parameter is blank.

> **Note:**
>
> The value you enter must be configured as an exact match of the event type expected in the SIP messages (except for the wildcard). Further, the value conforms to the event type BNF specified in RFC 3265.

7. **max-subscriptions**—Enter the maximum number of subscriptions allowed to a user for the SIP subscription event type you entered in the **event-type** parameter. Leaving this parameter set to 0 (default) means that there is no limit. You can set this parameter to a maximum value of 65535.
8. If you are entering multiple **subscribe-event** configurations, then you save them each by using the ACLI **done** command and then repeat Steps 6 and 7 to configure a new one. If you do not save each, then you will simply overwrite the first configuration repeatedly.

```
ORACLE(subscribe-event)# done
```

9. When you finish setting up **subscribe-event** configurations and have saved them, exit to return to the enforcement profile configuration.

```
ORACLE(subscribe-event)# exit
```

10. You also need to save the enforcement profile configuration.

```
ORACLE(enforcement profile)# done
```

Applying an Enforcement Profile to a Realm

For the Oracle USM to use the limits you have set up, you need to apply them to a realm.

To apply an enforcement profile to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

4. **enforcement-profile**—Enter the name of the enforcement profile you want to apply to this realm. This value corresponds to the **name** parameter in the enforcement profile configuration. This parameter has no default value.
5. Save and activate your configuration.

STUN Server

The Oracle USM supports RFC 3489, which defines Simple Traversal User Datagram Protocol (UDP) through Network Address Translators (NATs). Known as STUN, this lightweight protocol that allows applications to:

- Discover the presence and types of both NATs and firewalls between themselves and the public Internet
- Determine the public IP addresses allocated to them by the NAT

SIP endpoints use the STUN protocol to find out the public IP addresses and ports for SIP signaling and RTP media transport. Then they can use the address and port information to create multimedia sessions with other endpoints on the public network.

You can define STUN servers functionality on a per-realm basis, allowing you set up multiple STUN servers.

About STUN Messaging

STUN messages uses six messages, three of which are used for Binding and three of which are uses for the Shared Secret. While it supports all three Binding messages (request, response, and error), the Oracle USM does not support the Shared Secret Request or the message integrity mechanism that relies on the shared secret. When acting as a STUN server, the Oracle USM responds to STUN binding requests in accordance with RFC 3489 and the rfc3489bis draft.

STUN messages can contain the following attributes:

Message Type	Attribute Description
MAPPED-ADDRESS	Appears in the Binding Response; contains the source IP address and port from which the Binding Request was sent to the STUN server.
XOR-MAPPED-ADDRESS	Appears in the Binding Response; contains the MAPPED-ADDRESS information encoded in a way the prevents intelligent NAT devices from modifying it as the response goes through the NAT.
SOURCE-ADDRESS	Appears in the Binding Response; contains the IP address and port from which the STUN server sent its response.

Message Type	Attribute Description
CHANGED-ADDRESS	Appears in the Binding Response; contains an alternate STUN server IP address and port, different from the primary STUN server port. The STUN client might use this attribute to perform the NAT tests described in RFC 3489.
CHANGE-REQUEST	Appears in the Binding Request; instructs the STUN server to send its response from a different IP address and/or port. The STUN client might use this attribute to perform the NAT tests described in RFC 3489.
RESPONSE-ADDRESS	Appears in the Binding Request; defines an IP address and port to which the STUN server should send its responses. Appears in the Binding Request;
REFLECTED-FROM	Appears in the Binding Response; reflects the IP address and port from which a Binding Request came. Only included when the Binding Request has used the RESPONSE-ADDRESS attribute.
UNKNOWN-ATTRIBUTES	Appears in the Binding Error; reflects the mandatory attributes in a Binding Request message that the server does not support.
ERROR-CODE	Appears in the Binding Error; indicates an error was detected in the Binding Request, and contains an error code and reason phrase.

To perform NAT discovery, the endpoint (STUN client) sends a Binding Request to the STUN server port (IP address and port) with which it is configured. The STUN server then returns either a;

- Binding Response—Allows the transaction to proceed
- Binding Error—Halts the transaction, and prompts the client to take the action appropriate to the response given in the ERROR-CODE attribute

When the transaction proceeds and the STUN server sends the Binding Response, that response contains the MAPPED-ADDRESS attribute, which contains the IP address and port from which the server received the request. The STUN client then uses the MAPPED-ADDRESS when sending signaling messages.

For example, a SIP endpoint sends Binding Requests from its SIP port to determine the public address it should place in SIP headers, like the Via and Contact, of the SIP requests it sends. When this SIP endpoint prepares to make or answer a call, it sends Binding Requests from its RTP port to find out the public address it should place in SDP included in an INVITE request or response.

STUN Server Functions on the Oracle USM

When the Oracle USM receives a STUN message, it first determines its message type. Only STUN Binding Requests are processed, and all other message types are dropped without response.

Then the Oracle USM examines the Binding Request's STUN attributes. It returns error responses if it finds any unsupported mandatory attributes. This takes the form of a Binding Error Response, containing the ERROR-CODE attribute with reason 420 (Unknown Attribute) and an UNKNOWN-ATTRIBUTES attribute with a list of the unsupported attributes. If the Oracle USM receives a Binding Request with attributes that do not belong in STUN Binding Requests, it returns the Binding Error Response with the ERROR-CODE attribute with reason 400 (Bad Request).

Next the Oracle USM determines whether to follow RFC 3489 procedures or rfc3489bis procedures. If the Transaction ID contains the STUN cookie, then the Oracle USM follows rfc3489bis procedures; if not, the it follows RFC 3489 procedures. Because it defines the procedures for testing the NAT to see what type of NAT it is, RFC 3489 procedures are most complex. Issues with reliability of those results have caused testing procedures and attributes to be deprecated in fc3489bis.

RFC 3489 Procedures

The Oracle USM (the STUN server) constructs the Binding Response and populates it with these attributes:

- MAPPED-ADDRESS and (optionally) XOR-MAPPED-ADDRESS—Containing the source IP address and port from which the server saw the request come
- SOURCE-ADDRESS—Containing the IP address and port from which the server will send the Binding Response
- CHANGED-ADDRESS—Containing the STUN server port that has a different address and different port from the ones on which the server request was received

If the Binding Request contains a RESPONSE-ADDRESS attribute, the server adds the REFLECTED-FROM attribute with the IP address and port from which the server saw the request come. Then the server sends the Binding Response to the IP address and port in the RESPONSE-ADDRESS attribute. If the RESPONSE-ADDRESS attribute's IP address and port are invalid, the server sends a Binding Error Response with an ERROR-CODE attribute reason 400 (Bad Request) to the client.

If the Binding Request contains a CHANGE-REQUEST attribute, the server sends Binding Response from the IP address and port matching the information in the CHANGE-REQUEST. The following variations can occur:

- If the IP address and port flags are set, the server selects the server port with a different IP address and different port.
- If only the IP address flag is set, the server selects the server port with a different IP address but with the same port.
- If only the port flag is set, the server selects the server port with the same IP address but with a different port.

The selected server port appears in the Binding Responses's SOURCE-ADDRESS attribute. When there is no CHANGE-REQUEST attribute, the server uses the server port on which the Binding Request was received.

Finally, the server encodes the outgoing message and sends it to the client at either:

- The destination IP address and port in the REPONSE-ADDRESS attribute, if it was present in the Binding Request.
- The MAPPED-ADDRESS.

rfc3489bis Procedures

If the Binding Request contains the appropriate cookie in its Transaction ID, the server constructs a Binding Response populated with the XOR-MAPPED-ADDRESS attribute. That attribute will contain the source IP address and port from which the server saw the request come. Then the server encodes and sends the message to the client from the IP address and port

on which the request was received. The message is sent to the IP address and port from which the request came.

Monitoring

- STUN Server Statistics—You can display statistics for the STUN server using the ACLI **show mbcd stun** command when the STUN server has been enabled. However, if the STUN server has not been enabled since the last system reboot, the command does not appear and no statistics will be displayed.
- STUN Protocol Tracing—You can enable STUN protocol tracing two ways: by configuration or on demand.
  - By configuration—The Oracle USM's STUN protocol trace file is called stun.log, which is classified as a call trace. This means that when the system configuration's call-trace parameter is set to enabled, you will obtain STUN protocol information for the system. As with other call protocol traces, tracing data is controlled by the log-filter in the system configuration.
  - On demand—Using the ACLI **notify mbcd log** or **notify mbcd debug** commands, you enable protocol tracing for STUN. Using **notify mbcd debug** sets the STUN log level to TRACE. You can turn off tracing using the **notify mbcd onlog** or **notify mbcd nodebug** commands. Using **notify mbcd nodebug** returns the STUN log level back to its configured setting.

STUN Server Configuration

You configured STUN servers on a per-realm basis, one server per realm. To support that various NAT tests it describes, RFC 3489 requires that two different IP addresses and two different UDP port numbers be used for each server. So your STUN server will listen on a total of four STUN server ports. Although newer work does away with this requirement, the Oracle USMC supports it for the purpose of backwards compatibility.

For each realm configuration with an enabled STUN server, untrusted ACL entries will be added to forward all packets received on the four STUN Server Port.

To enable STUN server support for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **stun-enable**—Set this parameter to **enabled** to turn STUN server support for this realm on. This parameter defaults to **disabled**, meaning STUN server support is off.
5. **stun-server-ip**—Enter the IP address for the primary STUN server port. The default for this parameter is 0.0.0.0.

6. **stun-server-port**—Enter the port to use with the **stun-server-ip** for primary STUN server port. The default is 3478.
7. **stun-changed-ip**—Enter the IP address for the CHANGED-ADDRESS attribute in Binding Requests received on the primary STUN server port. This IP address must be different from than the one defined for the **stun-server-ip** parameter. The default for this parameter is 0.0.0.0.
8. **stun-changed-port**—Enter the port combination to define the CHANGED-ADDRESS attribute in Binding Requests received on the primary STUN server port. The default for this parameter is 3479.
9. Save and activate your configuration.

SIP ISUP Features

This section describes the Oracle USM's feaures for SIP ISUP.

SIP Diversion to SIP-ISUP Interworking

For networks in which there are devices that do not support SIP-T or SIP-I (and support native SIP alone), the Oracle USM now supports SIP Diversion interworking. This feature enables such devices to function properly in instances that require SIP-T/SIP-I style ISUP IAM message encapsulation in ISUP requests, and to receive any call forwarding information in the IAM according to ISUP standards.

The Oracle USM interworks a native SIP INVITE request to SIP-T one by inserting an ISUP IAM body based on the INVITE; this includes redirections information based on the Diversion header. This feature can also perform the reverse translation. That is, it can interwork a SIP INVITE that does have the ISUP IAM body to a non-ISUP INVITE. In this case, the Oracle USM generates the necessary Diversion headers based on the IAM's Redirection information.

SIP-ISUP Configuration

To use this feature, you set up:

- sip-profile—Defines the redirection behavior
- sip-isup-profile—Defines the ISUP version to use when supporting SIP-T

You can then apply these profiles to realms, session agents, and SIP interfaces where you want this feature enabled.

The **sip-profile** configuration contains information that defines redirection behavior for the configuration where you apply it. You can set the redirection behavior to: **none**, **isup**, or **redirection**. You also uniquely identify the profile so that you can apply it by name in other configurations. This is a multiple-instance configuration, meaning that you can set up as many SIP profiles as needed.

To set up a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-profile** and press Enter.

```
ORACLE(session-router)# sip-profile
ORACLE(sip-profile)#
```

4. **name**—Enter the name of the SIP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **redirection**—Choose the redirection mode you want to use: **none** (default), **isup**, or **redirection**. The **inherit** value is reserved for future use. Note that when you set this parameter to **isup**, you should configure along with it a SIP ISUP profile; this will avoid any possible incompatibility when support for this feature expands (as expected).
6. Save your work.

SIP ISUP Profile Configuration

To set up a SIP ISUP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-isup-profile** and press Enter.

```
ORACLE(session-router)# sip-isup-profile
ORACLE(sip-isup-profile)#
```

4. **name**—Enter the name of the SIP ISUP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **isup-version**—Specify the ISUP version you want used in this profile in order to support SIP-T: **ansi-2000** (default) or **itu-99**.
6. Save your work.

SIP-ISUP Configuration

When you want to enable this feature for a realm, session agent, or SIP interface, you configure the **sip-profile** and **sip-isup-profile** parameters with the name of the profile you want applied.

The sample here shows this feature being applied to a session agent, but the realm and SIP interface configurations also have the same two parameters you use to set up the feature.

To apply a SIP profile and a SIP ISUP profile to a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **session-agent** and press Enter.

```
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

4. **sip-profile**—Enter the name of the SIP profile, which defines the redirection behavior. This is the value you entered in the **name** parameter of the SIP profile configuration. This parameter has no default value.
5. **sip-isup-profile**—Enter the name of the SIP ISUP profile, which defines the ISUP version to use for SIP Diversion SIP-ISUP interworking. This is the value you entered in the **name** parameter of the SIP ISUP profile configuration. This parameter has no default value.
6. Save your work.

SIP-ISUP Format Version Interworking

ISUP message can be carried in SIP messages through either a standard body or through a multipart MIME encoded body. While ANSI and ITU are the two major groups, but each contains many specific variants. To facilitate instances where two sides of a call use different versions, the Oracle USM supports interworking between the following SIP ISUP formats: ANSI, ITU, ETSI-356 (an ITU variant), and GR-317 (an ANSI variant). To do so, the Oracle USM can move, delete, and add parameters to various sections of the message.

Details

The ISUP message version is determine by one of two things: the content type of the SIP message or the MIME content-type. When the base and version parameters do not match, the Oracle USM first uses the base parameter value to determine the format. If there is no base, the Oracle USM then checks the version parameter. And if there is neither, the Oracle USM uses the **isup-version** configured in the **sip-isup-profile** configuration from the inbound realm, session agent, or SIP interface. Available values for that parameter are **ansi-2000**, **itu-99**, **gr-317**, or **etsi-356**. The Oracle USM considers unknown any value for the version that fails to match one of these or is missing.

Messages that contain an unknown ISUP format pass through the Oracle USM untouched. If there are operations to be performed on them, however, SIP ISUP HMR will take place. After the body has been converted, the Oracle USM updates both the base and version parameters of the content-type.

Custom formats are not supported.

SIP-ISUP Format Interworking Configuration

This section show you how to set up a SIP-ISUP format interworking. First, you configure a SIP ISUP profile, and then you apply it to a realm, session agent or SIP interface.

To set up a SIP ISUP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-isup-profile** and press Enter.

```
ORACLE(session-router)# sip-isup-profile
ORACLE(sip-isup-profile)#
```

4. **name**—Enter the name of the SIP ISUP profile. You will use this name when you apply this profile to realms, session agents, and SIP interfaces. This parameter is required, and it has no default value.
5. **isup-version**—Specify the ISUP version you want to which you want to covert: **ansi-2000**, **itu-99**, **gr-317**, or **etsi-356**.
6. **convert-isup-format**—Set this parameter to **enabled** if you want to perform SIP ISUP format version interworking. The default is **disabled**, meaning that this feature is turned off.
7. Save your work.

SIP-ISUP Format Interworking Configuration

When you want to enable this feature for a realm, session agent, or SIP interface, you configure the **sip-isup-profile** parameter with the name of the profile you want applied.

The sample here shows this feature being applied to a session agent, but the realm and SIP interface configurations also have the same parameter you use to set up the feature.

To apply a SIP profile and a SIP ISUP profile to a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)#
```

2. Type **session-router** and press Enter.

```
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)#
```

3. Type **session-agent** and press Enter.

```
ACMEPACKET(session-router)# session-agent
ACMEPACKET(session-agent)#
```

4. **sip-isup-profile**—Enter the name of the SIP ISUP profile, which defines the ISUP version to convert to. This is the value you entered in the **name** parameter of the SIP ISUP profile configuration. This parameter has no default value.
5. Save your work.

HMR for SIP-ISUP

The Oracle USM's HMR functionality can operate on ISDN user party (ISUP) binary bodies. Using the same logic and mechanisms that are applied to SIP header elements, HMR for SIP-ISUP manipulates ISUP parameter fields and ISUP message parts. You can create MIME rules that function in much the same way the SIP header rules do; whereas SIP header rules can change the specific headers of a SIP message, MIME rules can manipulate targeted body parts of a SIP message.

RTN 1605

Changes and Additions to Equality Operators

The following table defines the additions and changes to HMR equality operators introduced with this feature.

Unlike the Boolean operators the ampersand (&) and the pipe (|), you can use the following equality operators in conjunction with string operators. For example, a header-value with its **comparison-type** set to **boolean**, can have this match-value evaluated: "($rule1.$elem1.$0 + $rule1.$elem2.@1) == $rule2.$0". Equality operators can also be uses with Boolean operators, as in this example: ($rule1.$0 == $rule2.$1) & $rue3.

Equality operators always evaluate to either true or false.

Equality Operator Symbol	Short Description	Detailed Information
==	String case sensitive quality operator	Performs a character-by-character, case-sensitive string comparison on both the left side and the right side of the operator.
~=	String case insensitive quality operator	Performs a character-by-character, case-insensitive string comparison on both the left side and the right side of the operator.
!=	String case sensitive not equal to equality operator	Performs a character-by-character, case-sensitive string comparison on both the left side and the right side of the operator, returning true if the left side is equal to or less than the right side of the operator.
>=	Greater than or equal to operator	Performs a string-to-integer conversion. If the string-to-integer comparison fails, the value is treated as 0. After the conversion, the operator will compare the two values and return true only if the left side is greater than or equal to the right side of the operator.
<	Less than operator	Performs a string-to-integer conversion. If the string-to-integer conversion fails, the value is treated as 0. After the conversion, the operator will compare the two values and return true only if the left side is less than the right side of the operator.
>	Greater than operator	Performs a string-to-integer conversion. If the string-to-integer conversion fails, the value is treated as 0. After the conversion, the operator will compare the two values and return true only if the left side is greater than the right side of the operator.

Reserved Words

To improve system performance and simply configuration, the Oracle USM now supports pre-defined reserved words for commonly-used URI parameters for HMR.

Reserved words retrieve values directly from the SIP message, without your needing to create rules to store them. Their function is similar to the $REMOTE_VIA_HOST and other already-defined variables. If the header or value does not exist in the SIP message, either an empty string is returned or—for Boolean uses—the value FALSE is returned.

Reserved words apply to these commonly-accessed SIP headers and their prefixes are:

- To—$TO_xxx
- From—$FROM_xxx

- Contact—$CONTACT_xxx
- Request URI—$RURI_xxx
- P-Asserted-Identity—$PAI_xxx
- P-Preferred-Identity—$PPI_xxx
- P-Called-Party-ID—$PCPID_xxx

The following table contains the list of supported reserved words and a description of each.

Reserved Word	Description
xxx_USER	The URI name of the header without any user parameters
xxx_PHONE	The URI user of the header as a phone number but without visual separators; may or may not contains a leading plus sign (+)
xxx_HOST	The URI host of the header
xxx_PORT	The URI port of the header; Value set to 5060 even if it is not actually in the message
CALL_ID	Resolves to the Call-ID of the current SIP message; added for convenience, and is a common store rule
TIMESTAMP_UTC	Timestamp is RFC 3339 format: 2009-10-10T22:00:09Z or YYY-MM-DDTHH:MM:SS:PPPZ. The .PPP refers to partial seconds and is optional; time is based on UTC.

The reserved word CRLF resolves to \r\n and is commonly used in MIME manipulation. If you are creating a new body, there might be a need for many CRLFs in the new-value parameter.

All of these operators cause additional overhead to the HMR processing because each operator requires an evaluation of the left and right sides of the expression. To speed up evaluation of new-value expressions, you can now enter escapable characters (\f, \n, \r, \t, \v) with a backslash (\) and the Oracle USM will covert them to escaped characters during the compilation of the expression (i.e., ACLI configuration time).

List of Reserved Words

The table below lists reserved, built-in variables for use in HMRs.

$ORIGINAL,LOCAL	$TRUNK_GROUP	$TARGET_PORT
$IP,LOCAL_PORT	$TRUNK_GROUP_CONTEXT	$MANIP_STRING
$REMOTE_IP	$REPLY_IP	$CRLF
$REMOTE_PORT	$REPLY_PORT	$TO_USER
$REMOTE_VIA_HOST	$TARGET_IP	$TO_PHONE
$TO_HOST	$TO_PORT	$FROM_USER
$FROM_PHONE	$FROM_HOST	$FROM_PORT
$CONTACT_USER	$CONTACT_PHONE	$CONTACT_HOST
$CONTACT_PORT	$RURI_USER	$RURI_PHONE
$RURI_HOST	$RURI_PORT	$PAI_USER
$PAI_PHONE	$PAI_HOST	$PAI_PORT
$PPI_USER	$PPI_PHONE	$PPI_HOST
$PPI_PORT	$PCPID_USER	$PCPID_PHONE
$PCPID_HOST	$PCPID_PORT	$CALL_ID
$TIMESTAMP_UTC	$MANIP_PATTERN	$T_GROUP

$T_CONTEXT,	$M_STRING	$M_PATTERN

Reserved words that do not use the referenced header name include:

- $TARGET_* references the To-URI hostname
- $REPLY_* references the FROM-URI hostname

Reserved word names that are equal, to support legacy version configurations, include:

- $T_GROUP is the same as TRUNK_GROUP
- $T_CONTEXT is the same as TRUNK_CONTEXT
- $M_STRING is the same as MANIP_STRING (and refers to another manipulation)

Reserved variables operate just like user-defined variables. The recommended usage syntax is "$variable.$0". Use of the $0 suffix causes the HMR to retrieve the value of the data. If the $0 is omitted then the resulting value would be <TRUE/FALSE>.

For example, "$PAI_USER" is a valid usage. In this case, the TRUE/FALSE result could be used to determine if there was not a username in the PAI header, or that no PAI header was present in the message.

Changes to Action

In releases prior to S-C6.2.0, the **sip-manip** action is only supported in the header rule (**header-rule**) configuration. This limitation has been deemed unnecessary, and so you can now set the action parameter to **sip-manip** at all levels of HMR configuration, including element rules.

About MIME Rules

MIME rules (set up in the ACLI **mime-rules** configuration) operate much the same way that SIP header rules do. You can set parameters in the MIME rules that the Oracle USM uses to match against specific SIP methods and message types. The system compares the search criteria against the body or body parts using the type of comparison you choose. Offering a variety of selection, you can pick kind of manipulation that suits your needs; the Oracle USM then takes action with matching and new values to change the SIP message.

> **Note:**
>
> when you use the **delete** action on a multi-part MIME string that reduces a number of bodies down to one, the SIP message remains a multi-part MIME message with only one body part (and thereby avoids the header conflicting with the message itself).

You identify the MIMe rule by configuring a content type that refers to the specific body part on which to operate. For example, given a SIP Content-Type header with the value multipart/mixed;boundary=unique-boundary-1, you would enter a content-type value of **application/sdp** to manipulate specifically on the SDP portion of the SIP message. The Oracle USM knows automatically if it is operating on SIP messages with single or multiple body parts, and the content-type setting applies to both kinds. And when making its comparison, the Oracle USM matches the content-type of the body with regard to case (case insensitive), ignoring any header parameters.

Both for making comparisons against the body part and for new/replacement values, the Oracle USM treats the match and new values you set for a MIME rule as ASCII strings. Therefor, a mime rule operating on a binary body part will yield an improper conversion of a new value with respect to the binary body part.

Within MIME rules, you configure MIME headers, which operate on the specific headers in the match body part of the SIP message. The Oracle USM uses the MIME header name to run a string comparison to match the specific header in the message's body part.

Using these rules, you can also manipulate the preamble—or the SIP message text that follows the headers but precedes the body separator. To do so, enter the keyword **@preamble** for the content type parameter in the MIME rule. Likewise you can manipulate the epilogue—or the text that follows the last body part after the last separator—using the keyword **@epilogue**.

Note that the ACLI limits character entries to 255 characters before the return character must be entered, but MIME parts can easily exceed this 255-character size. So you might need to enter a value larger that 255 characters. To do so, you start your entry (in the match-value or new-value parameters) with a plus sign (+). The plus sign instructs the system to add the string after it to the pre-existing match or new value. For the new-value parameter, the Oracle USM checks the value immediately for validity. Be sure that when you are appending values to a new-value that the entire expression is valid at each point where strings are appended.

MIME Rules Configuration

This section shows you how to configure MIME rules and MIME headers.

To configure MIME rules:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter. **If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. **Type mime-rules and press Enter.**

   ```
   ORACLE(sip-manipulation)# mime-rules
   ORACLE(sip-mime-rules)#
   ```

5. **name**—Enter a name for this MIME rule. This parameter is required and has no default.

6. **content-type**—Enter the content type for this MIME rule. This value refers to the specific body part in the SIP message body that is to be manipulated. For example, given a SIP Content-Type header with the value multipart/mixed;boundary=unique-boundary-1, you would enter a content-type value of **application/sdp** to manipulate specifically on the SDP portion of the SIP message.

   To manipulate the SIP preamble or epilogue, enter the keyword **@preamble** or keyword **@epilogue**.

7. **action**—Choose the type of action you want to be performed: **none**, **add**, **delete**, **manipulate**, **store**, **sip-manip**, and **find-replace-all**. These are the same actions you can select when configuring SIP header manipulation. The default is **none**.
8. **comparison-type**—Enter the way that you want body part of the SIP message to be compared. This choice dictates how the Oracle USM processes the match rules against the SIP header. the default is **case-sensitive**. The valid values are: **case-sensitive**, **case-insensitive**, **boolean**, **refer-case-sensitive**, **refer-case-insensitive**, and **pattern-rule**.
9. **msg-type**—Enter the SIP message type on which you want the MIME rules to be performed. Valid values are **any**, **request**, and **reply**. The default value is **any**.
10. **methods**—Enter the list of SIP methods to which the MIME rules applies. There is no default for this parameter.
11. **match-value**—Enter the value to match against the body part in the SIP message. This is where you can enter values to match using regular expression values. Your entries can contain Boolean operators.
12. **new-value**—When the action parameter is set to **add** or to **manipulate**, enter the new value that you want to substitute.

    To configure MIME headers for performing HMR operations on specific headers in the matched body part of the SIP message:
13. Follows Steps 1 through 4 above.
14. **Type mime-header-rules and press Enter.**

    ```
    ORACLE(sip-mime-rules)# mime-header-rules
    ORACLE(sip-mime-header-rules)#
    ```
15. **name**—Enter a name for this MIME header rule. This parameter is required and has no default.
16. **mime-header**—Enter the value to be used for comparison with the specific header in the body part of the SIP message. There is no default for this parameter.
17. **action**—Choose the type of action you want to be performed: **none**, **add**, **delete**, **manipulate**, **store**, **sip-manip**, and **find-replace-all**. The default is **none**.
18. **comparison-type**—Enter the way that you want the header in the body part of the SIP message to be compared. This choice dictates how the Oracle USM processes the match rules against the SIP header. the default is **case-sensitive**. The valid values are: **case-sensitive**, **case-insensitive**, **boolean**, **refer-case-sensitive**, **refer-case-insensitive**, and **pattern-rule**.
19. **match-value**—Enter the value to match against the header in the body part of the SIP message. This is where you can enter values to match using regular expression values. Your entries can contain Boolean operators.
20. **new-value**—When the action parameter is set to **add** or to **manipulate**, enter the new value that you want to substitute.
21. Save your work.

About MIME ISUP Manipulation

MIME ISUP manipulation supports performing HMR operations on SIP ISUP binary bodies, and is configured in the **mime-isup-rule** configuration. This configuration works the same way that the MIME rule configuration does and contains the same parameters for you to set, but it also includes additional parameters and a sub-configuration targeted specifically for ISUP application.

Oracle USM MIME ISUP Parameters

- **isup-msg-type**—Refers to specific ISUP message types (such as IAM and ACM). The Oracle USM uses with the msg-type parameter (which identifies the SIP message) in the matching process. You enter values in this parameters as a list of numbers rather than as an enumerated value because of the large number of ISUP message type, and the range is between 0 and 255.
- **isup-spec**—Specifics how the Oracle USM is to parse the binary body; valid values are the enumerated type. The values for this parameter are these SIP ISUP formats
  - ANSI-2000—Corresponding to ANSI T1.113-2000
  - ITU-99—Corresponding to ITU Q.763
    Because ISUP messages do not identify their format, you must designate which you want to use.
- **isup-parameter-rules** (sub-configuration)—If you are familiar with HMR, then think of this parameter as being similar to the element-rule for a SIP header rule. You use it to create, manipulate, and store different parameters in the body of the ISUP message. Two parameters for this rule are unique: **parameter-rule** and **parameter-format**.
  - **parameter-rule**—Using ISUP parameter mapping, this setting identifies which of the ISUP parameters on which your want to perform manipulation. This parameter takes values between 0 and 255, and you must know the correct ISUP mapping value for your entry. The Oracle USM calculates the offset and location of this parameter in the body. Note that the value returned from the body does not the type or length, only the parameter value. For example, a parameter-type value of 4 acts on the Called Party Number parameter value.
    In accordance with the ISUP specifications, only certain message types are allowed to have optional parameters. And if optional parameters are present, an offset field must exist for them; so its value is 0 even if there are no optional parameters in the SIP message. For example, if you define a SIP ISUP rule that applies to all message types and adds a parameter that is neither fixed nor variable, The system adds it as an optional parameter regardless of whether that message type should not support optional parameters.

    If you define an ISUP parameter rule with an **add** action and an empty **new-value**, the Oracle USM uses the default for that parameter. If you define an ISUP parameter rule with a **replace** action and no parameters exist, the Oracle USM will not perform any action. This behavior is consistent with that of SIP header rules in that a value can only be replaced if it already exists. If there is a value and no new value is set, the Oracle USM set it as a zero-length parameter.
  - **parameter-format**—This parameter converts the specific parameter to a string representation of that value. Valid values for **parameter-format** are: **number-param**, **hex-ascii** (default), **binary-ascii**, **ascii-string**, and **bcd**.
    Both match and new values are encoded and decoded by the designated parameter-format type. In this regard, the **match-value** decodes the parameters and the **new-value** encodes the ASCII string into the respective binary format.

    Note if you enter a new-value setting larger than the size of the parameter, the Oracle USM will perform no operation and will generate a corresponding error log message.

    The following table provides information about the values you can enter:

parameter-format Setting	Description
hex-ascii	Default. Converts the entire binary body. Non-hexadecimal characters fail in matching against the body part if they are in the match-value setting and non-hexadecimal characters places in a new-value setting result in no operation being formed.
binary-ascii	Converts each hexadecimal value to its corresponding string, binary representation. For example, the Oracle USMwould convert the ISUP parameters with a hexadecimal binary value of 8A to 10001010. Non-binary digit characters fail when matching against the body part if they are contained in the match-value setting and non-binary characters in the new-value setting results in no operation being performed.
ascii-string	Treats the binary parameter as true ASCII in raw format. The Oracle USM supports only the printable range of ASCII characters. If a value in the ISUP parameters cannot be decoded to ASCII, the Oracle USM returns an empty string. Non-printable or meta characters cannot be entered as new-value settings, so this presents no issues for encoding.
bcd	In ISUP speak, BCD refers to the binary forma of the number used as a half a byte nibble, with the byte's lower nibble containing the first digit and the higher containing the second digit. For example, the number 127 is encoded as the two binary bytes 0x2107 on the wire. Using this mode, the Oracle USM treats the binary ISUP content as BCD; it should decode it from 0x2107 to the string 1270, and from a string of 127 it should decode it as 0x2107. Since a byte has two nibbles, a nibble might have to be added. And when the Oracle USM performs decoding, it cannot know that a BCD byte represents one or two ASCII digits—so it assumes there are two. The number-param setting decodes the parameter as a common number parameter. The Oracle USM sees the odd/even bit as in the first bite as telling it how many nibbles to decode correctly, and it will set the odd-even when it decodes. Non-binary digit characters fail to match against the body part if they are contained in the match value, and non-binary characters in the new value results in no operation being performed.
number-param	As the decimal value of the specified number type, treats the parameters as a generic number parameter type. For example, a parameter-type 4 acts on the Called Party Number parameter. When the action type is replace or add, the Oracle USM automatically sets the parameter's odd-even bit based on the number being inserted in relation to the new-value setting. If the Numbering Plan Indicator bits are 0b001 (ISDN, E.164), then the Oracle USM sets the Nature of Address field to 0b0000100 (international). If this number type is added to a non-existent parameter field, then the Numbering Plan Indicator field is 0b0000011 (national number). If this number type is added to a non-existent parameters field, then the Numbering Plan Indicator field will be set to 0b001 (ISDN,E.164) and the Oracle USMwill also follow the previous rules. Regardless of the action type you set, the string represented for match-value use for this type will be the numbers of the address fields after the BCD coding. There will be a leading plus sign (+) if the Number Plan is 0b001 and the Nature of Address is 0b0000100 ((international); otherwise, there will not be a plus sign (+). If it cannot convert the data field to a number parameter, the Oracle USM will return an empty string. And if the new-value is not in digit form or cannot fit in the specified parameter type field, the Oracle USM takes no action.

Adding an ISUP Body to a SIP Message

Unlike the MIME manipulation you can use by setting the SIP header rules accordingly, you can add MIME parts to SIP messages using the MIME rules configuration.

You can configure a SIP header manipulation to add an ISUP body to a SIP message. and the Oracle USM adds them after any SDP parts if they are present. You can add an ISUP body to a SIP message in two ways:

- You can create a **mime-isup-rule** with the **action** type set to **add**, and enter the entire body in string hexadecimal form in the **new-value** parameter.
- You can leave the **new-value** parameter empty at the **mime-isup-rule** level and create an add rule for an **isup-param-rule**.
  In this case, the Oracle USM creates the corresponding ISUP message based on the **isup-msg-type** value and supply all of the parameters with their default values. Since the **isup-msg-type** takes a list of values as a valid entry, for this case it only uses the first one. However, the Oracle USM ignores the **isup-msg-type** value if you set the **new-value** parameter. And the **isup-param-rule**, if configured, overwrite the default value or add a new parameter based on the defined parameter type.

  It is also possible that you might supply a **new-value** both at the **mime-isup-rule** level and at the **isup-param-rule** level. If you do, the **new-value** entry from the **mime-isup-rule** is parsed into an ISUP object and the **isup-param-rule** operates on that object.

MIME ISUP Manipulation Configuration

This section shows you how to configure MIME ISUP manipulation.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-manipulation** and press Enter. **If you are adding this feature to an existing configuration, then remember you must select the configuration you want to edit.**

   ```
   ORACLE(session-router)# sip-manipulation
   ORACLE(sip-manipulation)#
   ```

4. **Type mime-isup-rules and press Enter.**

   ```
   ORACLE(sip-manipulation)# mime-isup-rules
   ORACLE(sip-mime-isup-rules)#
   ```

5. **name**—Enter a name for this MIME ISUP rule. This parameter is required and has no default.

6. **content-type**—Enter the content type for this MIME rule. This value refers to the specific body part in the SIP message body that is to be manipulated. For example, given a SIP Content-Type header with the value multipart/mixed;boundary=unique-boundary-1, you would enter a content-type value of **application/sdp** to manipulate specifically on the SDP portion of the SIP message.

   To manipulate the SIP preamble or epilogue, enter the keyword **@preamble** or keyword **@epilogue**.

7. **action**—Choose the type of action you want to be performed: **none**, **add**, **delete**, **manipulate**, **store**, **sip-manip**, and **find-replace-all**. These are the same actions you can select when configuring SIP header manipulation. The default is **none**.
8. **comparison-type**—Enter the way that you want body part of the SIP message to be compared. This choice dictates how the Oracle USM processes the match rules against the SIP header. the default is **case-sensitive**. The valid values are: **case-sensitive**, **case-insensitive**, **boolean**, **refer-case-sensitive**, **refer-case-insensitive**, and **pattern-rule**.
9. **msg-type**—Enter the SIP message type on which you want the MIME rules to be performed. Valid values are **any**, **request**, and **reply**. The default value is **any**.
10. **methods**—Enter the list of SIP methods to which the MIME rules applies. There is no default for this parameter.
11. **match-value**—Enter the value to match against the body part in the SIP message. This is where you can enter values to match using regular expression values. Your entries can contain Boolean operators.
12. **new-value**—When the action parameter is set to **add** or to **manipulate**, enter the new value that you want to substitute.
13. **isup-spec**—Specify how the Oracle USM is to parse the binary body; valid values are the enumerated type. The values for this parameter are these SIP ISUP formats:
    - **ANSI-2000** (default)—Corresponding to ANSI T1.113-2000
    - **ITU-99**—Corresponding to ITU Q.763
14. **isup-msg-type**—Identify the specific ISUP message types (such as IAM and ACM) on which to operate. The Oracle USM uses with the **msg-type** parameter (which identifies the SIP message) in the matching process. You enter values in this parameters as a list of numbers rather than as an enumerated value because of the large number of ISUP message type, and the range is between 0 and 255. There is no default for this parameter.
15. **mime-header**—Enter the value to be used for comparison with the specific header in the body part of the SIP message. There is no default for this parameter.

    To configure ISUP parameters rules:
16. Follows Steps 1 through 4 above.
17. **Type isup-parameter-rules and press Enter.**

    ```
    ORACLE(sip-mime-isup-rules)# isup-param-rules
    ORACLE(sip-isup-param-rules)#
    ```
18. **name**—Enter a name for this ISUP parameter rule. This parameter is required and has no default.
19. **mime-header**—Enter the value to be used for comparison with the specific header in the body part of the SIP message. There is no default for this parameter.
20. **action**—Choose the type of action you want to be performed: **none**, **add**, **delete**, **manipulate**, **store**, **sip-manip**, and **find-replace-all**. The default is **none**.
21. **comparison-type**—Enter the way that you want the header in the body part of the SIP message to be compared. This choice dictates how the Oracle USM processes the match rules against the SIP header. the default is **case-sensitive**. The valid values are: **case-sensitive**, **case-insensitive**, **boolean**, **refer-case-sensitive**, **refer-case-insensitive**, and **pattern-rule**.
22. **match-value**—Enter the value to match against the header in the body part of the SIP message. This is where you can enter values to match using regular expression values. Your entries can contain Boolean operators.

23. **new-value**—When the action parameter is set to **add** or to **manipulate**, enter the new value that you want to substitute.
24. **parameter-type**—Using ISUP parameter mapping, enter which of the ISUP parameters on which your want to perform manipulation. This parameter takes values between 0 and 255, and you must know the correct ISUP mapping value for your entry. The Oracle USM calculates the offset and location of this parameter in the body. Note that the value returned from the body does not the type or length, only the parameter value. For example, a parameter-type value of 4 acts on the Called Party Number parameter value.
25. parameter-format—Enter how you want to convert specific parameter to a string representation of that value. Valid values for **parameter-format** are: **number-param**, **hex-ascii** (default), **binary-ascii**, **ascii-string**, and **bcd**. Both match and new values are encoded and decoded by the designated parameter-format type. In this regard, the **match-value** decodes the parameters and the **new-value** encodes the ASCII string into the respective binary format.
26. Save your work.

Configuration Example

This section provides an example of a SIP manipulation configuration that shows MIME rules and MIME ISUP rules.

```
sip-manipulation
        name                            manip
        description
        header-rule
                name                            headerRule1
                header-name                     Date
                action                          add
                comparison-type                 case-sensitive
                msg-type                        reply
                methods
                match-value
                new-value
                element-rule
                        name                            elemRule1
                        parameter-name
                        type                            header-value
                        action                          add
                        match-val-type                  any
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       "August 19, 1967"
mime-rule
                name                            mimeRule1
                Content-Type                    application/SDP
                action                          manipulate
                comparison-type                 case-sensitive
                msg-type                        request
                methods
                match-value
                new-value
                mime-header
                        name                            mimeHeaderRule1
                        mime-header-name                Content-Disposition
                        action                          add
comparison-type                 case-sensitive
                        match-value
```

```
                        new-value                       "signal;
handling=required"
        mime-isup-rule
                name                            mimeRule1
                content-type                    application/ISUP
                action                          manipulate
                comparison-type                 case-sensitive
                msg-type                        request
                methods                         INVITE
                match-value
                new-value
                isup-spec                       {ansi00, itu-92}
                isup-msg-type                   0 (0-256 IAM, ACM, etc.)
                mime-header
                        name                            mimeHeaderRule1
                        mime-header-name                Content-Disposition
                        action                          add
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       "signal;
handling=optional"
                isup-param-rule
                        name                            isupRule1
                        parameter-type                  # {0-256 specific type)
                        parameter-format                {number-parameter, hex,
binary, ascii, bcd}
                        action                          add
                        comparison-type                 case-sensitive
                        match-value
                        new-value                       "signal;
handling=optional"
```

SIP Session Timer Feature

SIP does not have a keepalive mechanism for established sessions and it does not have the capability of determining whether a session is still active. User agents (UAs) may be able to determine whether a session has timed out by using session specific mechanisms, but proxies cannot always determine when sessions are still active.

The Oracle USM provides a SIP session timer feature that, when enabled, forwards the re-INVITE or UPDATE requests from a User Agent Client (UAC) to a User Agent Server (UAS) in order to determine whether or not a session is still active. This refresh feature works for both UAs and proxies. The following paragraphs provide additional information about the session timers on the Oracle USM.

How the Session Timer Feature Works

During an active SIP call session, when a UA fails to send a BYE message at the end of the session, or when the BYE message gets lost due to network problems, the proxy cannot determine when the session has ended. Therefore, the proxy may hold onto resources associated with the call session for indefinite periods of time causing the session to never time out.

The SIP session timer feature adds the capability to periodically refresh SIP sessions by sending repeated INVITE (re-INVITE) or UPDATE Session Refresh Requests. These requests are sent during active call legs to allow UAs or proxies to determine the status of a SIP session. The Session Refresh Requests along with the session timers determine if the active sessions stay active and completed sessions are terminated.

When you enable the session timer feature on the Oracle USM, it periodically sends out a Session Refresh Request (re-INVITE or UPDATE). The Response that is returned to the Oracle USM contains a success status code (2xx) that contains a session timer interval. The Oracle USM then refreshes the session timer each time it receives the 2xx Response containing that session timer interval.

The initial INVITE message sent from the UAC to the UAS contains two fields that make up the session timer interval in the SIP Session Header:

- Session-Expires (SE) - Specifies the maximum amount of time, in seconds, that can occur between session refresh requests in a dialog before the session is considered timed out.
- Minimum-SE (Min-SE) - Specifies the minimum allowed value, in seconds, for the session expiration.

The following displays the session timer interval values inserted in the SIP session INVITE message per RFC 4028:

```
INVITE sip:9109621001@192.168.200.99 SIP/2.0
Via: SIP/2.0/UDP 192.168.200.49:5060;branch=z9hG4bK0g6t23200gd0res4l580.1
Max-Forwards: 69
From: <sip:rick@192.168.1.48>;tag=SDr08od01-188c3fbc-b01a-4d68-b741-09e5dc98a064
To: sip:149@192.168.1.49
Contact: <sip:rick@192.168.200.49:5060;transport=udp>
Call-ID: SDr08od01-9c12b48e3b0f7fad39ff3a2e0ced5ed3-v3000i1
CSeq: 3941 INVITE
Allow: INVITE, ACK, BYE, CANCEL, UPDATE, PRACK
Supported:  timer
Session-Expires: 1800
Min-SE: 90
Content-Type: application/sdp
Content-Length: 236

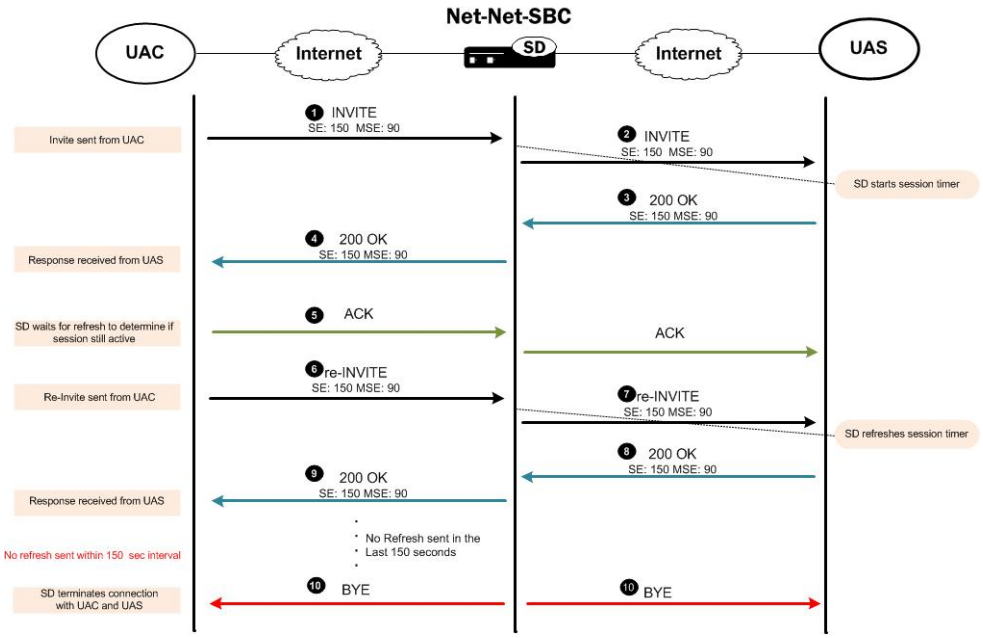
v=0
o=- 3462189550 3462189550 IN IP4 192.168.200.49
s=pjmedia
c=IN IP4 192.168.200.49
t=0 0
m=audio 20000 RTP/AVP 0 8 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=sendrecv
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

If the Oracle USM receives an INVITE from the UAC with a Session-Expires header, it starts a new session timer, or refreshes an existing session timer and then forwards the INVITE to the UAS. The subsequent 2xx Responses and re-INVITES also include the session timer intervals. If the Oracle USM does not receive a session refresh within the time specified in the session timer interval, the session timer expires, and the Oracle USM terminates the session between the UAC and the UAS.

The following occurs when you enable the session timer feature on the Oracle USM:

1. The UAC sends an INVITE to the Oracle USM with the SE and min-SE values (session timer interval). The Oracle USM starts a session timer.
2. The Oracle USM forwards the INVITE to the User Agent Server (UAS) with the same values.
3. The UAS sends the 200 OK Response to the Oracle USM with the session interval values.

4. The Oracle USM forwards the Response to the UAC.
5. The UAC sends an ACK (Acknowledge) to theOracle USM, and the Oracle USM forwards the ACK to the UAS.
6. The UAC sends out a re-INVITE (Session Refresh Request) to the Oracle USM with the session interval values. The Oracle USM refreshes the session timer.
7. The Oracle USM forwards the re-INVITE to the UAS.
8. The UAS sends the 200 OK response to the Oracle USM with the session interval values.
9. The Oracle USM sends the 200 OK response to the UAC.
10. If the Oracle USM does not receive a Response within the session interval (150 seconds in the following illustration), the timer expires, and the Oracle USM terminates the session between the UAC and the UAS. The following illustration shows an example of a dialog between the UAC, the Oracle USM, and the UAS during an active session.



When the Oracle USM terminates a session it sends a BYE to both the ingress and egress call legs. If accounting is configured, the Oracle USM also sends a RADIUS stop record with Acct-Terminate-Cause = Session-Timeout. You can enable or disable the use of the session timers using the ACLI interface at session-router > sip-config > options.

SIP Session Timer Configuration

You can configure the session timer feature on the Oracle USM to periodically refresh SIP sessions and determine whether or not a session is still active. If the Oracle USM determines that a session is no longer active, it terminates the session based on the session timer interval settings.

To configure the session timer feature on the Oracle USM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router-related configurations.

```
ORACLE(configure)# session-router
```

3. Type **sip-config** and press Enter to access the SIP config-related configurations. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. Enter options followed by the following value:

   - +session-timer-support

     ```
     ORACLE(sip-config)# options +session-timer-support
     ```

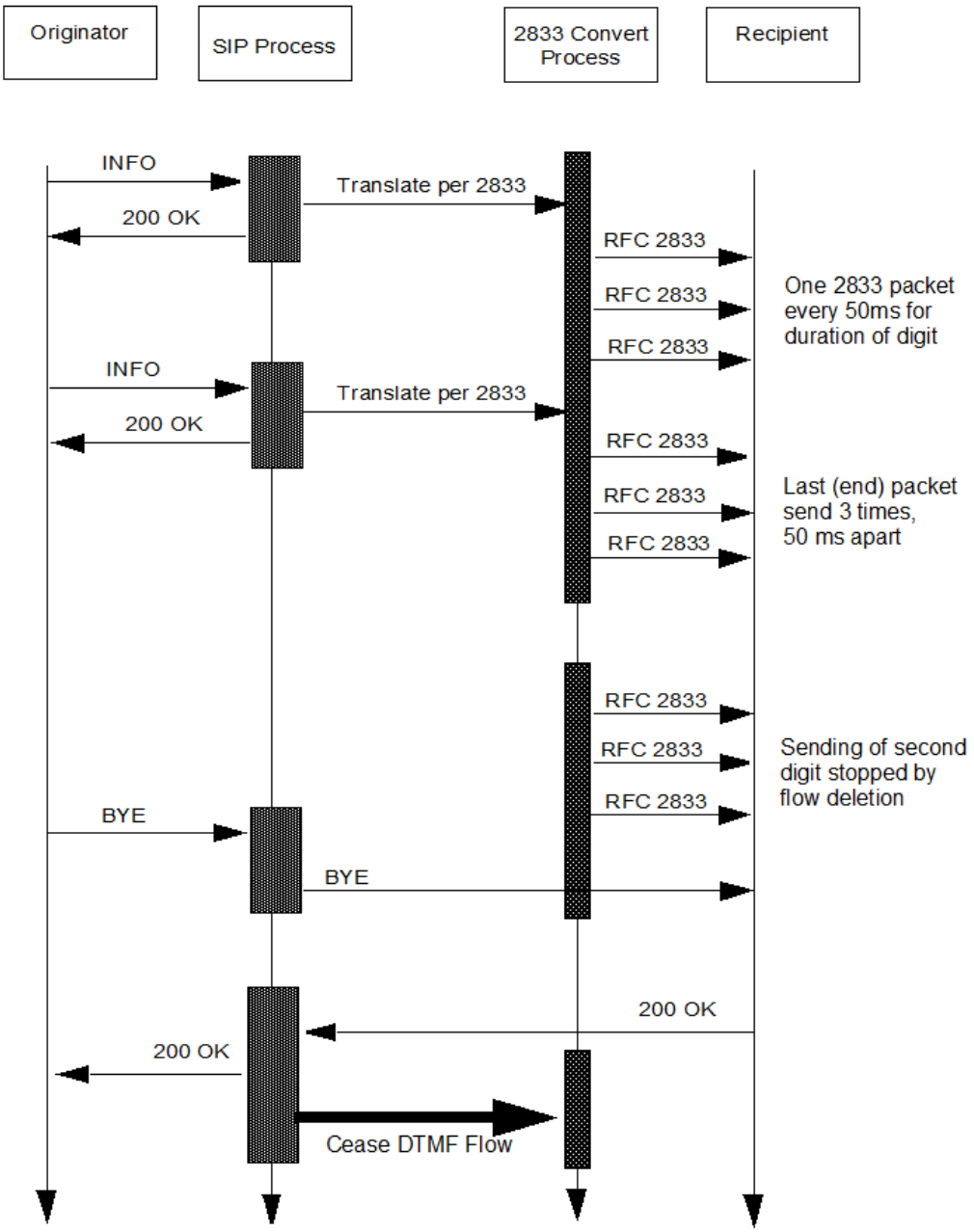
     This value enables the system to start the session timer for session refreshes coming from the UAC. The system determines whether or not a session is active based on session refreshes or responses. It terminates the session when no session refreshes occur within the session timer interval. To disable the session timer feature, enter options followed by **-session-timer-support**.

     ```
     Oracle USM(sip-config)# options -session-timer-support
     ```

     > **Note:**
     >
     > To disable session timers, you must add a minus sign (-) before the session-timers-support value.

DTMF Conversion Processing

Release S-CX6.3F1 provides a configurable SIP option to implement DTMF-to-RFC2833 tone translation. The current, default implementation, which performs well in most network topologies is shown below.



When the Oracle Oracle USM receives an INFO DTMF request, the SIP process determines whether or not it needs to perform DTMF-to-RFC2833 translation. If translation is required, the process forwards the DTMF to the 2833 convert process for translation and transmission to the recipient. Immediately after off-loading the DTMF, the SIP process sends a 200 OK response for the INFO. As shown in the figure, the 2833 convert process generates a number of RFC2833 packets to represent received DTMF digits.

Specifically, the 2833 convert process generates one RFC 2833 packet every 50 milliseconds for the duration of the DTMF digit, whose length is specified in the INFO request, and two retransmits of the last packet (known as the end packet) 50 milliseconds apart.

Consequently, the time interval between the 200 OK and the actual transmission of the RFC 2833 translation is the sum of the DTMF duration and 100 ms.

> **Note:**
>
> This time interval can be shortened to 100 ms by enabling the rfc2833-end-pkts-only-for-non-sig parameter in media-manger which results in SD only generating the last packet and its two retransmits.

The early 200 OK allows the endpoint to send the next DTMF digit before the SD has sent all the RFC2833 packets, resulting in the next digit being queued internally by the 2833 convert process before being sent.

A problem arises if the SIP process receives a BYE request from the DTMF originator while queued digits are awaiting translation and transmission. In such an event, the SIP process immediately forwards the BYE request to the recipient, ending the session with DTMF digits awaiting translation and transmission.

An alternative DTMF conversion model provides for a feedback mechanism from the 2833 convert process to the SIP process. With this model enabled, the SIP process buffers a received BYE until it obtains confirmation that all queued DTMF digits have been translated and transmitted. Only after obtaining confirmation, does it forward the BYE to terminate the session.

This processing model is enable by a SIP option, **sync-bye-and-2833**, and requires that **rfc2833-mode** parameter on the egress interfaces is NOT set to dual , any value other than dual , is supported.

1. From superuser mode, use the following command sequence to access sip-config configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

2. Use the SIP option **sync-bye-and-2833** to delay BYE processing until DTMF-to-RFC2833 translation has been completed.

   ```
   ORACLE(sip-config)# options +sync-bye-and-2833="enabled"
   ORACLE(sip-config)#
   ```

3. Use the **done** and **exit** commands to complete configuration.

   ```
   ORACLE(sip-config)# done
   ORACLE(sip-config)# exit
   ORACLE(session-router)#
   ```

SIP-KPML to RFC 2833 Conversion for DTMF Events

The Oracle USM supports SIP KPML to RFC 2833 conversion for DTMF events.

SIP KPML to RFC 2833 Negotiation

The Oracle USM removes the Allow-Event header if it contains the value KPML.

A SUBSCRIBE request is sent for each KPML event upon a successful invite.

After the SUBSCRIBE even is successful, the Oracle USM will process all NOTIFY requests and respond with 200 OK messages.

The SDP-Insertion feature will be used if the Access-side does not offer a SDP in the invite and the core side does not accept an offer-less invite.

RFC 2833 to SIP KPML Negotiation

The Oracle USM adds an Allow-Event header with the value KPML to the INVITE message.

If a SUBSCRIBE message is received from the Access side, the Oracle USM will respond with a 200 OK message.

A KPML NOTIFY request is sent for each 2833 DTMF entered, until the subscription is terminated.

KPML-2833 Interworking on a SIP Interface Configuration

To configure KPML - 2833 interworking on a SIP interface:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **kpml-interworking**—Set this parameter to enabled to use KPML-2833 interworking.
4. Type **done** to save your configuration.

LMSD SIP Call Progress Tone Interworking

The Oracle USM supports Legacy Mobile Station Domain interworking tht allows SIP interworking with User Agents that support the 3GPP2 LMSD. The LMSD provides support for existing Mobile Stations in a network that supports IP bearers.

LMSD uses Alert-Info headers in a 180 Ringing response to an INVITE to indicate to a User Agent specific tones to generate locally by the UAC. Most User Agents that do not support LMSD will ignore the SDP in the Alert-Info and will not play ringback locally. The lmsd-interworking option allows for the system to suppress SDP in 180 Ringing, 486 Busy Here, and 503 Service Unavailable responses, so that the UAC plays local ringback.

There are no additional license requirements.

LMSD Interworking Configuration

You can apply **lmsd-interworking** to the sip-interface facing the LMSD User Agents. For Example, if the endpoints uspporting LMSD are located in the core realm, then the **lmsd-interworking** option would be added to the core **sip-interface**.

To enable the LMSD Interworking option:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter. If you are adding this feature to a pre-existing configuration, you will need to select and edit it.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing options, a Space, the option name **lmsd-interworking** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +lmsd-interworking
   ```

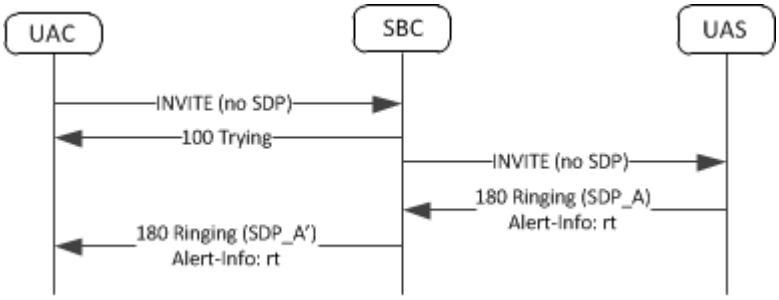
   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save your work.

LMSD Offerless INVITE handling

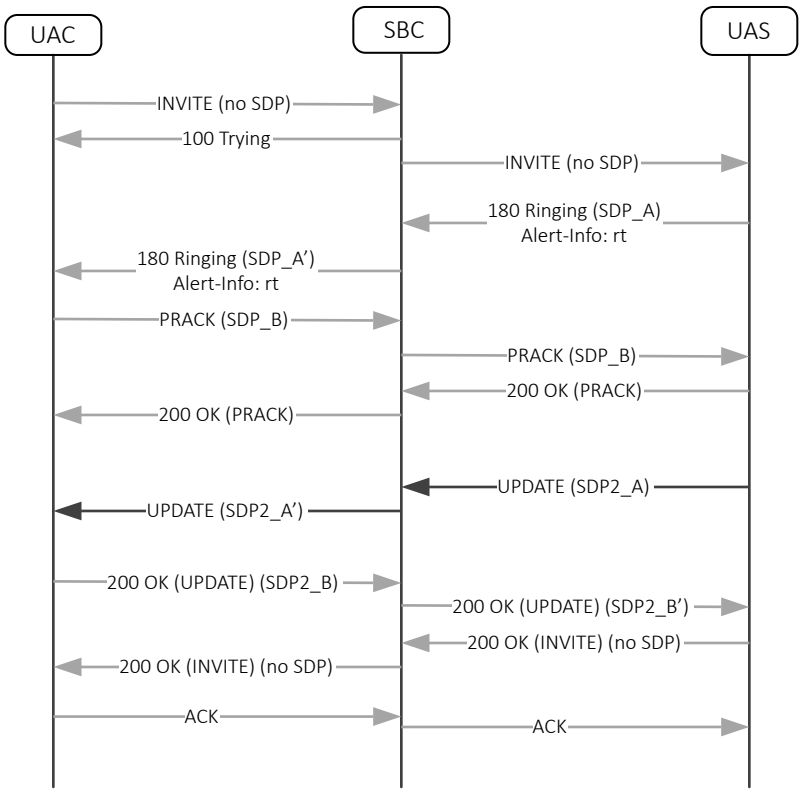
To enhance LMSD interworking, the Oracle USM does not remove SDP from a 180 response sent back to the UAC when the initial request did not contain SDP. The Oracle USM also forwards UAS-side UPDATE requests to the UAC; it does not respond locally. These represent behavioral changes and require no configuration.

When LMSD Interworking is configured on a SIP interface, and when the UAS includes SDP in the 180 Ringing message (and the Alert-Info header is set to rt), the Oracle USM no longer strips the SDP when forwarding the 180 back to the UAC.



When LMSD interworking is configured on a SIP interface, and in the case that an early dialog has been established (a dialog where the final 2xx class response to INVITE request has not yet arrived), some call flows include an UPDATE later in the call. The Oracle USM now forwards

the Update message, SDP included, directly to the UAC, whereas before it responded to the UAS's UPDATE locally.



SIP re-INVITE Suppression

Some of the Interactive Voice Response (IVR) systems that support SIP frequently change the media transport address (IP address and/or port number) when switching between voice menus and/or prompts by sending a re-INVITE.

Often, no other parameters or properties of the session are changed in these re-INVITEs. The frequent re-INVITEs can create performance and capacity problems in other systems along the path of the IVR system and the caller's User Agent.

You can configure the **suppress-reinvite** option on your Oracle USM, allowing it to store the previous INVITE and its 200-OK response. Having this information allows the system to reply locally when a re-INVITE that changes only the media transport addresses is recieved.

No license requirements to enable this feature.

SIP re-INVITE Suppression Configuration

To enable SIP re-INVITE Suppression:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **suppress-reinvite** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(sip-interface)# options +suppress-reinvite
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the SIP interface configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

RFC 4028 Session Timers

The Oracle USM supports RFC 4028 Session Timers. In this role, it acts as a B2BUA between two endpoints and then enforces the timer values on each call leg independently. The RFC 4028 abstract states:

This document defines an extension to the Session Initiation Protocol (SIP). This extension allows for a periodic refresh of SIP sessions through a re-INVITE or UPDATE request. The refresh allows both user agents and proxies to determine whether the SIP session is still active. The extension defines two new header fields:

- Session-Expires—which conveys the lifetime of the session
- Min-SE—which conveys the minimum allowed value for the session timer

The following parameters in the session-timer-profile configuration element are used for this feature:

session-expires—The value of the session expires header in seconds

min-se—The value of the Min-SE header in seconds (this is a minimum session expires value)

force-reinvite—Sets if the Oracle USM will send a reINVITE to refresh the session timer when applicable.

request-refresher—Set on the outbound side of a call what the Oracle USM sets the refresher parameter to. Valid values are uac, uas, or none.

response-refresher—Set on the inbound side the value of the refresher parameter in the 200OK message. Valid values are uac or uas.

In this section, the notion that a UAC or UAS supports Session Timers is indicated by the presence of the Supported: timer header and option tag.

Ingress Call Leg

Setting 200 OK's Session-Expire value

The session timer is based on the negotiation between each side's session expires value. The final value on the ingress leg is returned by the Oracle USM to the UAC, unless there is an error.

The Oracle USM can always reduce the session expires value it returns to the UAC. It checks that the Session-Expires: header is larger than the SIP Interface's min-se value. The Oracle USM then compares the received Session-Expires: header to the configured session-expires configuration element and uses the lower value for the 200 OK's Session-Expires: header. If this outbound Session-Expires: value is lower than the received Min-SE: header, it will be bumped up to the Min-SE: header's value.

If the Oracle USM's Min-SE value is larger than the Session-Expires: header, a 422 (Session Interval Too Small) message is returned to the UAC containing the Oracle USM's configured Min-SE value.

When the UAC supports (but does not require) Session Timers and the Oracle USM does not support session timers, a 200 OK is returned to the UAC with no indication of session timer support.

Refresher

The initial UAC, the side that sends the INVITE, can set itself to be the refresher (uac) or the Oracle USM as the refresher (uas). Whoever is the refresher is indicated in the 200 OK. If the UAC does not specify any refresher, the Oracle USM uses it's response-refresher value in the 200 OK. If that value is set to uas, the Oracle USM creates and sends a re-INVITE toward the UAC with previously negotiated session expiration values.

Once the Oracle USM becomes the refresher, it does not relinquish that role. Then, when the Oracle USM sends refresh requests, it does not change any parameters (refresher role & timers) from the initial request negotiation.

UAC does not Support Session Timers

If the UAC's initial request does not include a Session-Expires: header, then the 200 OK will include the **session-timer-profile > session-expires** value on the ingress leg in the Session-Expires: header.

The Oracle USM also inserts the refresher parameter as configured. The orientation of UAC/UAS on the Oracle USM's view of a call leg can change if later in the call flow the endpoint designates the Oracle USM as the refresher.

> **Note:**
>
> When the request doesn't support Session Timers, the Oracle USM's reply adds session timer support according to configuration.

If the Oracle USM receives a message with a Require: timer header, and the inbound SIP interface or the final UAS do not support session timers, a 420 (Bed Extension) is returned to the UAC.

Egress Call Leg

Outbound INVITE Message

When the Oracle USM's outbound interface is configured with session timers, it forwards an INVITE to the UAS with the following headers:

Session-Expires— Oracle USM inserts the outbound SIP interface's session-timer parameter

Session-Expires refresher parameter— Oracle USM inserts the request-refresh parameter

Min-SE— Oracle USM inserts the outbound SIP interface's session-timer parameter

Supported—Supported header has the timer option tag

> **Note:**
>
> Require/Proxy-Require—If the timer parameter is present in the Require or Proxy-Require: header field in the request received from the UAC, it will be removed.

No Session Timer Configuration

If the ingress SIP interface supports session timers, and the original INVITE from the UAC included session timer support, the INVITE request sent to the UAS will have no session timer support. However, the Supported: timer header will be created. This ensures that the Oracle USM does not get a 421 response for 'timer' from the UAS.

If the ingress SIP interface supports session timers, and the UAC's initial INVITE did not include session timer support, then the INVITE sent to the UAS will have no session timer support (headers) as well.

If the ingress SIP interface does not support session timers, the INVITE is forwarded with no Session Timer alteration.

UAS Initial Response

Upon receiving a 200 OK from the UAS, if the response specifies uac as the refresher, the 200 OK includes a Session-Expires header and specifies uac as the refresher, the Oracle USM will assume the refresher role. If the 200 OK does not include a Session-Expires header, and the egress interface supports session timers, then the Oracle USM assumes the refresher role.

UAS Returns Errors

422 Session Interval Too Small—The Oracle USM in response sends the request again with new values in the 'Session-Expires' header field based on the 'Min-SE' value present in the 422 response.

421 Extension Required for 'timer'—This response can only happen if none of the other three entities (UAC, ingress SIP interface and egress SIP interface) support session timers. The 421 is forwarded through the system to the original UAC.

420 Bad Extension for 'timer'—This response should never happen because the Oracle USM will never send Require: timer header. But the event this error is received, it will be forwarded to the original UAC.

Session Refreshes

On either side of the call, the Oracle USM can be responsible for initiating the session refreshes or responding to the session refreshes.

Oracle USM as Refresher

The Oracle USM sends the refresh request when half the session expiration has elapsed. The Oracle USM always wants to remain the refresher and maintain the initially agreed upon session expiration timers.

Creating the Refresh Message

The refresh message takes the form of a re-INVITE when the force-reinvite parameter in the session timer profile is enabled. If this parameter is disabled and the remote end supports UPDATE requests, an UPDATE message will be sent.

UPDATE messages contain no SDP information.

Re-INVITE messages contain the SDP that is the same as what was sent before.

The refresh request's Session-Expires: header value is set to the existing value for the session. The refresher parameter is set to uac since the Oracle USM acts like a UAC for this refresh transaction. The Min-SE header is also included.

Processing the Refresh Response

The session expires value in the 2xx response is accepted and the timer restarts.

If the remote end does not include any session expiration parameters, the Oracle USM continues to support session timers, and assumes that the refresh interval is the same as before.

Any response that is 422 Session Interval Too Small is handled as expected. The Oracle USM resends the refresh request again with new values based on the 422 response. Any other response to the refresh request that is not a dialog/usage destroying response is treated like a 200 OK response.

Subsequent refresh requests are created and sent after half the previous refresh interval. If non-2xx, dialog / usage destroying responses are received, the Oracle USM reduces the following refresh intervals by half, as long as the final interval is not less than 32 seconds. The Oracle USM then uses this period for sending refresh requests until it successfully receives a 2xx response.

Oracle USM as Refresh Responder

Processing the Refresh

The refresh request is processed similarly to the initial request regarding the session timer parameters. The session timer for this call leg is restarted when the Oracle USM when it sends the 200 OK response for the refresh request.

Forwarding the Refresh

When the Oracle USM receives an UPDATE request, it is forwarded to the other end since the Oracle USM cannot determine whether this request is only for session refreshing, or for other purposes as well.

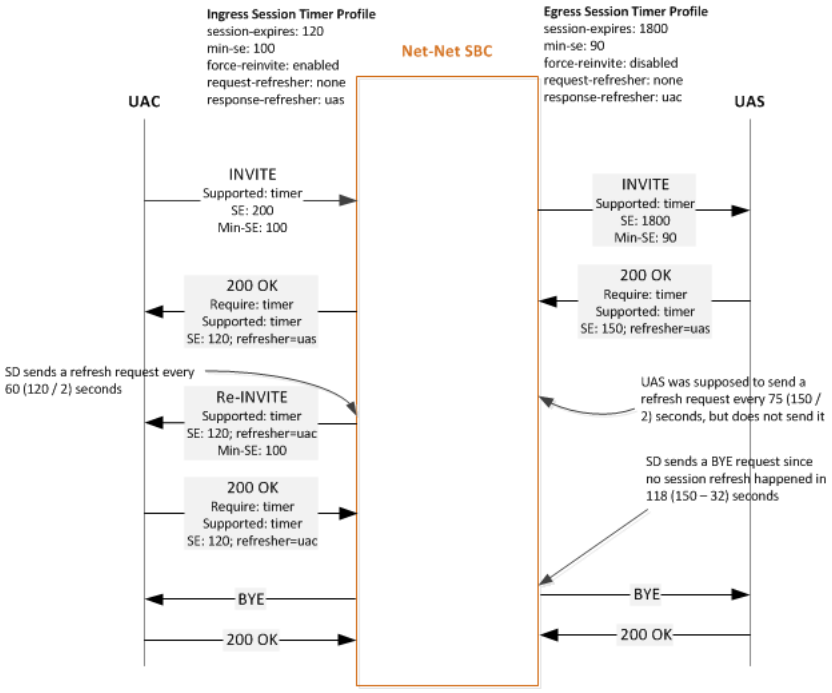
When the Oracle USM receives a re-INVITE request, it will determine whether this request needs to be suppressed, or should it be forwarded to the other end.

Timer Expiration

If the Oracle USM fails to receive a session refresh request before the session expiration, the session will be terminated before the full session time. This is computed according to:

$$time = period - min(\frac{period}{3}, 32)$$

After the real expiration time elapses, the Oracle USM sends a BYE request in both directions to terminate the session.



Interaction with SIP Features

Consider the following sections that have interactions with RFC 4028 Support.

sip-config option session-timer-support

A configured session-timers-profile on a SIP interface overrides the session-timer-support option in the SIP config. The Oracle USM can still act in proxy mode for some calls and B2BUA for other calls considering which SIP interfaces session timer profiles are not configured for.

sip-feature Support

When the UAC sends a Require: timer header is in the initial request and the Oracle USM does not support session timers, and no sip-feature configuration element is configured for 'timer' for that realm, the Oracle USM replies with a 420 (Bad Extension) response for 'timer'.

When the UAC sends a Require: timer header is in the initial request and the Oracle USM does support session timers, the timer tag is removed from the Require: header even if a sip-feature configuration element is configured for 'timer'. This also applies for the Proxy-Require: header.

Oracle recommends you do not configure a sip feature configuration element while using the Session Timers feature.

sip-interface option suppress-reinvite

SIP re-INVITE suppression is automatically enabled for a SIP interface when session timers are enabled. This behavior prevents re-INVITEs whose purpose is only for session refreshes from being forwarded to the other call leg. The first re-INVITE received from a UAS on the terminating call leg will be passed to the UAC on the originating call leg since the Oracle USM has no prior INVITE request coming from the UAS to match against.

Re-INVITEs are suppressed only when the Oracle USM receives the same INVITE request back-to-back without any intervening re-INVITE request in the opposite direction, or an UPDATE or PRACK request in either direction.

> **Note:**
>
> The SIP reINVITE Supression parameters are not replicated on the standby system in an HA environment. The first re-INVITE after a switchover will be forwarded to the far end.

Examples

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
1	INVITE → Supported: timer SE: 200 ———————————— ← 200 OK Require: timer SE: 200; refresher=uas	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas this element becomes refresher	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	INVITE→ Supported: timer SE: 500 Min-se: 400 ———————————— ← 200 OK Require: timer SE: 400; refresher=uas
2	INVITE → Supported: timer SE: 1200; refresher=uas ———————————— ← 200 OK Require: timer SE: 500; refresher=uas	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uac this element becomes refresher	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 ———————————— ← 200 OK

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
3	INVITE → Supported: timer SE: 1200; refresher=uac Min-se: 800 ——————————— ← 200 OK Require: timer SE: 800; refresher=uac	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas	No session timer configuration this element becomes refresher	INVITE → Supported: timer ——————————— ← 200 OK Require: timer SE: 400; refresher=uac
4	INVITE → Supported: timer SE: 200; refresher=uac ——————————— ← 200 OK Require: timer SE: 200; refresher=uac	session-expires: 500 min-se: 200 request-refresher: none response-refresher: uas	No session timer configuration	INVITE → Supported: timer ——————————— ← 200 OK
5	INVITE → Supported: timer SE: 200 ——————————— ← 200 OK	No session timer configuration	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 ——————————— ← 200 OK
6	INVITE → Supported: timer SE: 200 ——————————— ← 200 OK Require: timer SE: 400; refresher=uas	No session timer configuration SBC behavior stays same as current behavior	No session timer configuration	INVITE → Supported: timer SE: 200 ——————————— ← 200 OK Require: timer SE: 400; refresher=uas
7	INVITE → Require: timer SE: 200 ——————————— ← 420 Unsupported: timer	No SIP feature for timer No session timer configuration SD behavior stays same as current behavior	No session timer configuration	

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
8	INVITE → Require: timer SE: 200 ——————— ——— ← 420 Unsupported: timer	SIP feature configured for timer No session timer configuration SD behavior stays same as current behavior	No session timer configuration	INVITE → Required: timer SE: 200 ——————— ——— ← 420 Unsupported: timer
9	INVITE → Require: timer SE: 200 ——————— ——— ← 200 OK	SIP feature configured for timer No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas this element becomes refresher	INVITE → Supported: timer SE: 500 Min-se: 400 ——————— ——— ← 200 OK Require: timer SE: 500; refresher=uac
10	INVITE → Require: timer SE: 200 ——————— ——— ← 200 OK Require: timer SE: 200; refresher=uac	No SIP feature for timer session-expires: 500 min-se: 200 request-refresher: none response-refresher: uac	session-expires: 500 min-se: 400 request-refresher: uas response-refresher: uas	INVITE → Supported: timer SE: 500; refresher=uas Min-se: 400 ——————— ——— ← 200 OK Require: timer SE: 500; refresher=uas
11	INVITE → Require: timer SE: 200 ——————— ——— ← 420 Unsupported: timer	No SIP feature for timer No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	
12	INVITE → SE: 200 ——————— ——— ← 200 OK SE: 500; refresher=uas	session-expires: 500 min-se: 500 request-refresher: none response-refresher: uas this element becomes refresher	No session timer configuration	INVITE → ——————— ——— ← 200 OK

Ex	Messages on Originating Call Leg	Ingress SIP Interface Config	Ingress SIP Interface Config	Messages on Terminating Call Leg
13	INVITE → ←— 200 OK	No session timer configuration	session-expires: 500 min-se: 400 request-refresher: none response-refresher: uas	INVITE → Supported: timer SE: 500 Min-se: 400 ←— 200 OK Require: timer SE: 400; refresher=uas
14	INVITE → ←— 421 Require: timer	No session timer configuration SD behavior stays same as current behavior	No session timer configuration	
15	INVITE → Supported: timer SE: 200 ←— 422 Min-se: 400	session-expires: 500 min-se: 400		
16	INVITE → Supported: timer SE: 200 ←— 200 OK Require: timer SE: 200; refresher=uac	session-expires: 800 min-se: 90 request-refresher: none response-refresher: uac	session-expires: 800 min-se: 90 request-refresher: none response-refresher: uac	INVITE → Supported: timer SE: 800 Min-se: 90 ←— 422 Min-se: 900 INVITE → Supported: timer SE: 900 Min-se: 900 ←— 200 OK Require: timer SE: 900; refresher=uas

RADIUS Interim record Generation

When refresh requests (UPDATE or Re-INVITE) are sent by the Oracle USM , no RADIUS Interim records are generated because session parameters do not change when these requests are sent.

When UPDATE requests are received by the Oracle USM , no RADIUS Interim records are generated.

When Re-INVITE requests are received by the Oracle USM , RADIUS Interim records are generated if the generate-interim parameter is enabled.

ACLI Configuration

To configure a session timer profile object:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **session-timer-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-timer-profile
   ORACLE(session-timer-profile)#
   ```

4. **name**—Enter a name for this session timer profile.
5. **session-expires**—Enter the session timer value in seconds you wish this object to use natively.
6. **min-se**—Enter the minimum session timer value in seconds for this object.
7. **force-reinvite**—Leave the default of enabled for the Oracle USM to always use reINVITEs for session refreshes. Set this parameter to disabled for the Oracle USM to try using UPDATEs for session refreshes.
8. **request-refresher**—Set this to the value to insert in the refresher parameter in the Session-Expires: header on the originating call leg that the Oracle USM includes in the 200 OK response message. Valid values are **uac** and **uas**.
9. **response-refresher**—Set this to the value to insert in the refresher parameter in the Session-Expires: header on the terminating call leg that the Oracle USM includes in the INVITE message. Valid values are **uac**, **uas**, **none**.
10. Type **done** to save your work and continue.

    To apply a session timer profile to a SIP interface:

11. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

12. Type **session-router** and press Enter.

    ```
    ORACLE(configure)# session-router
    ORACLE(session-router)#
    ```

13. Type **sip-interface** and press Enter.

    ```
    ORACLE(session-router)# sip-interface
    ORACLE(sip-interface)#
    ```

    If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

14. **session-timer-profile**—Enter the name of a session timer profile object you have configured and want applied to this SIP interface.
15. Type **done** to save your work and continue.

Verify Config Validation

The Oracle USM's verify-config function checks that the value configured in all sip-interfaces' session-timer-profiles correspond to a configured session-timer-profile name. The following is be generated when this check fails:

```
ERROR: sip-interface [id] has reference to session-timer-profile [xyz] which
does not exist
```

show sipd status

The show sipd status command now contains new statistic, called Refreshes Sent which reflects the number of refresh requests that the Oracle USM has sent. For example:

```
ORACLE#show sipd status
SIP Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Sessions               0       1       0          2       1       1
Subscriptions          0       0       0          0       0       0
Dialogs                0       2       0          4       2       2
CallID Map             0       2       0          4       2       2
Rejections             -       -       0          0       0
ReINVITEs              -       -       0          2       1
ReINV Suppress         -       -       0          1       1
Media Sessions         0       1       0          2       1       1
Media Pending          0       0       0          0       0       0
Client Trans           0       3       2         10       3       3
Server Trans           0       0       0          6       3       3
Resp Contexts          0       0       0          6       3       3
Saved Contexts         0       0       0          0       0       0
Sockets                2       2       0          2       2       2
Req Dropped            -       -       0          0       0
Refreshes Sent         0       0       0          0       0       0
DNS Trans              0       0       0          0       0       0
DNS Sockets            0       0       0          0       0       0
DNS Results            0       0       0          0       0       0
Rejected Msgs          0       0       0          0       0       0
```

305 Response to Registrations on Secondary Interfaces

Certain devices are provisioned with point of contact (POC) lists for registration. In the context of geographic redundancy, if a UE is unable to register with its primary POC, it proceeds to its secondary POC. It is desirable to designate certain Oracle USM as secondary in order to redirect the UE to re-register with the primary SBC.

This feature is designed for Oracle USMs sitting behind Oracle Communications Subscriber-Aware Load Balancers (SLBs). This feature allows users to designateUSMs sitting behind SLBs as secondary. When registered endpoints in a realm attempt to register with the primary POC, and registration fails, the UEs proceed to their secondary point of contact (POC). Once connection to the primary POC has been restored, users can execute a command to offload the UE registrations in a given realm with a 305 response to the UEs in order for the UEs to re-

register with the next POC. When the scenario consists of two POCs, the endpoint re-registers with the primary POC.

You enable this feature with an option through the SIP interface of the secondary SBC. The **notify sipd offload-users <realm>** command marks endpoints associated with a given realm to receive a 305 message. Once this command is executed, any subsequent requests from UEs in this designated realm receive a 305 message. Once a 305 message is issued, the registration cache is cleared to allow the UE request to be forwarded to the primary SBC. If the UE re-registers again with the secondary SBC, it will not receive a second 305 message. The **notify sipd offload-users <realm>** command must be executed again in order to repeat the process.

The **show registration** command is expanded to include a counter for the number of registrations received on secondary interfaces, as well as the number of endpoints marked to receive a 305.

SNMP support for this feature tracks the total count of registrations on secondary interfaces. The **apSipSecInterfaceRegThresholdExceededTrap** is generated when the total number of registrations on secondary interfaces exceeds the configured threshold, and **apSipSecInterfaceRegThresholdClearTrap** is generated when the total count falls below the configured threshold.

ACLI Instructions and Examples

To enable this feature, you must select an option for the SIP-interface of the secondary Oracle USM:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ACMEPACKET(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(session-router)# sip-interface
   ACMEPACKET(sip-interface)#
   ```

4. **options**—Type **secondary** to designate this system as the secondary point of contact.

   ```
   ACMEPACKET(session-router)# options secondary
   ```

5. Save and activate your configuration.

notify sipd offload-users

The **notify sipd offload-users <realm>** command marks the endpoints associated with the given <realm> to receive a 305 message. Once this command is executed, any re-registrations received from an endpoint in that realm receives a 305 message.

A confirmation message is sent if the command succeeds:

```
Realm <realm> is marked to offload users at HH:MM:SS
```

If the given <realm> is not a secondary realm, the following message is displayed:

```
Realm <realm> is not secondary
```

If the given <realm> does not exist, the following message is displayed:

```
Realm <realm> is not found
```

> **Note:**
>
> This command is only valid for secondary systems. If the secondary option is not enabled for theOracle USM, this command produces an error message.

show registration

The **show registration** command displays the total number of endpoints associated with secondary interfaces, and the total number of endpoints marked to receive a 305 response.

> **Note:**
>
> The total number of SIP calls to receive a 305 response are included in the Registrations marked for 305 counter.

The total number of SIP calls associated with secondary realms are included in this count.

```
AcmePacket# show registration
11:02:13-102
SIP Registrations          -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
User Entries           1       1       1          1       1       1
...
Secondary Interface Registrations=NNN
Registrations marked for 305=MMM
access1: N1
access2: N2
```

show registration sipd

The **show registration sipd** command includes the following parameters:

show registration sipd sec-by-ip—displays the registrations sent to secondary interfaces by IP address.

```
Registration Cache                    MON FEB 06 2012  16:33:49
                                                        Num
  IP Address      User                                  Contacts Registered at
--------------- ----------------------------- -------- ---------
. . . .
```

show registration sipd sec-by-user—displays the registrations sent to secondary interfaces by user name.

```
Registration Cache                    MON FEB 06 2012  16:33:49
                                                        Num
  Phone Number    User                                  Contacts Registered at
--------------- ----------------------------- -------- ------------
. . . .
```

show sipd endpoint-ip

If the registered endpoint is associated with a secondary SIP interface, the show sipd endpoint-ip command displays SIP Interface: secondary.

```
# show sipd endpoint-ip 1
User <sip:12341@172.16.101.67>
  Contact exp=290
    UA-Contact: <sip:12341@172.16.26.1:5060> UDP keep-acl
         realm=net172 local=172.16.101.67:5060 UA=172.16.26.1:5060
    SD-Contact: <sip:12341-2k86jbp9bj925@192.168.101.67:5060> realm=net192
    Call-ID: 1-839@172.16.26.1'
    SA=192.168.26.2
    Service-Route='<sip:h19216806003.dg.com;lr>'
  SIP Interface: secondary
```

SNMP Configuration

To configure the threshold for the number of registrations on secondary SIP interfaces in order to generate a trap:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ACMEPACKET(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

4. **options**—Type **sec-reg-threshold =X,** where X equals the number of registrations received on all secondary interfaces. Once that number has been reached, the apSipSecInterfaceRegThresholdExceededTrap is generated.

   ```
   ACMEPACKET(session-router)# options secondary
   ```

5. **options**—Type **sec-reg-threshold-clear =Y.** When the number of registrations received on all secondary interfaces drop below Y, the apSipSecInterfaceRegThresholdExceededClearTrap is generated.

   > **Note:**
   >
   > The value of sec-reg-threshold-clear =Y must be less than sec-reg-threshold =. The values cannot be equal.

   ```
   ACMEPACKET(session-router)# options secondary
   ```

6. **options**—Type **sec-reg-threshold-alarm-severity** = [minor | major | critical] to set the alarm severity for APP_ALARM_SEC_INTF_REG_EXCEED.

7. Save and activate your configuration.

SNMP

The apSipSecInterfaceRegThresholdExceededTrap trap is generated when the total count of all registrations on the secondary interfaces cross **sec-reg-threshold-exceed** is exceeded.

Trap Name	Object Identifier Name: Number	Description
Object Identifier Name: apSipInterfaceRegNotificationsGroup		
apSipSecInterfaceRegThresholdExceededTrap	.1.3.6.1.4.1.9148.3.15.2.1.2.0.1	The trap will be generated if the total number of registrations on all secondary SIP interfaces exceed threshold.
apSipSecInterfaceRegThresholdClearTrap	.1.3.6.1.4.1.9148.3.15.2.1.2.0.2	The trap will be generated if the total number of registrations on all secondary SIP interfaces go below clear threshold.

An alarm is also generated for this event. The severity of the alarm is configured in the **sec-reg-threshold-alarm-severity** option.

Alarm Name	Alarm ID	Alarm Severity	Cause(s)	Example Log Message	Trap Generated (Trap Reference)
Hardware Alarms					
APP_ALARM_SEC_INTF_REG_EXCEED	0X50018	Minor by default. The severity is configurable.	The total number of registrations on all secondary SIP interfaces exceeds the configured threshold.	Number of Secondary Interface registrations <total> has exceeded the secondary interface registration threshold of <max>.	apSyslogMessageGenerated (ap-slog.mib) apEnvMonStatusChangeNotification (ap-env-monitor.mib) apSysMgmtFanTrap (ap-smgmt.mib)

The following objects monitor the number of registrations on secondary interfaces:

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSipInterfaceRegObjectsGroup		
apSipSecInterfaceTotalRegistrations	.1.3.6.1.4.1.9148.3.15.1.1.1.1.0	The total number of registrations on all secondary SIP interfaces.
apSipSecInterfaceRegThreshold	.1.3.6.1.4.1.9148.3.15.1.1.1.2.0	The max threshold for registrations on all secondary SIP interfaces beyond which trap and alarm will be raised
apSipSecInterfaceClearThreshold	.1.3.6.1.4.1.9148.3.15.1.1.1.3.0	The threshold for registrations on all secondary SIP interfaces below which if alarm was raised before, it will be cleared.

12

Number Translation

About Number Translation

Oracle USM number translation is used to change a layer-5 endpoint name according to prescribed rules. Number translations can be performed on both the inbound and the outbound call legs independently, before and after routing occurs. Number translation is used for SIP, H.323, and SIP/H.323 interworking configurations.

Number translation takes place twice for both H.323 and SIP calls. The first number translation is applied to the incoming leg of the call, before the outgoing route is selected. The second number translation is applied to the outgoing leg of the call after the outgoing route is selected.

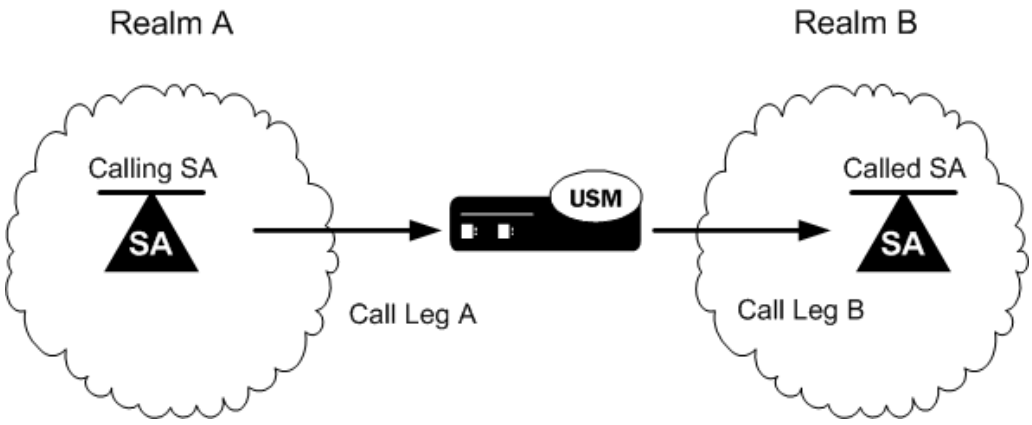
Number translation can be used to strip address prefixes added by external gateways. It can also be used to add a string tag to an address in order to implement a local policy routing scheme, and then remove the tag upon egress from the Oracle USM. The most common use of number translation is to add or remove a “1” or a + from a phone number sent from or addressed to a device.

Number Translation Implementation

Oracle USM number translations are implemented in three steps. First, the individual number translation rules are defined in the translation rules subelement. Next, the established rules are grouped in a specified order to apply to calling and called numbers. This second step occurs in the session translation element. Finally, session translations are attached to either session agents or realms in the session agent element or realm configuration element.

Number translations attached to session agents take precedence over number translations attached to realms. If no number translation is applied to a session agent, then the Oracle USM will use the number translation applied to a realm. If a number translation is applied to both a realm and session agent, the translation attached to the session agent will apply. If session agents and realms have no associated translations, then all numbers will remain in their original forms as they pass through the Oracle USM.

Within each realm or session agent, the number translation is applied to either the incoming or outgoing call leg. This distinction between incoming and outgoing calls is made from the point of view of the Oracle USM. The following diagram illustrates the number translation concept.



The following table shows you which parameters to apply a session translation ID in order to affect the corresponding leg of the call as shown in the illustration.

Leg	Calling SA	Called SA	Realm A	Realm B
A	IN Translation ID		IN Translation ID	
B		OUT Translation ID		OUT Translation ID

Number Translation in SIP URIs

Number translations only change the user portion of the URI. A typical SIP URI looks like **sip:user@hostname**. The user portion can take the form of either a phone number or any other string used for identification purposes.

Within the SIP header exists a Request URI, a To URI, and a From URI. The session translation element's rules calling parameter modifies the From URI, while the rules called parameter modifies the Request URI and the To URI.

Number Translation Configuration Overview

Configuring the number translation feature requires the following steps:

1. Configure individual translation rules in the translation rules element.
2. Group these rules for use in the session translation element.
3. Apply these groups of rules on a per session agent or per realm basis using the appropriate fields in the session agent or realm configuration elements.

Translation Rules

The translation rules subelement is where the actual translation rules are created. The fields within this element specify the type of translation to be performed, the addition or deletion to be made, and where in the address that change takes place. Translations are not applied to any realm or session agent in this element.

When creating translation rules, first determine the type of translation to perform. The following table lists and describes the three types of number translations.

Field Value	Description
add	This translation type adds a character or string of characters to the address.
delete	This translation type deletes a character or string of characters from the address.
replace	This translation type replaces a character or string of characters within the address. Replace works by first applying the delete parameter then by applying the add parameter.

After you set the translation type, you define the string to add or delete. The wildcard term for a string to delete is the at-sign, @. Finally, you specify the character position in the address to make the addition or deletion.

The character position where an add or delete occurs is called an index. The index starts at 0 (immediately before the leftmost character) and increases by 1 for every position to the right you move. In order to specify the final position in an address, use the dollar-sign, $.

To create a translation rule that deletes a string:

Translation Rules for Deleting Strings

To create a translation rule that deletes a string:

1. Enter a descriptive name for this translation in the ID field.
2. If you are deleting a specific string, enter it in the delete string field.
3. If you are deleting a portion of the address string, enter the index number in the delete index field. For this type of deletion, remember to enter the number of characters you are deleting in the form of at-signs @ in the delete string field.

   The first matched string will be deleted, any remaining strings that match will remain. For example, if the address is 187521865 and the string to delete is “18,” only the first instance of 18 will be deleted. The second instance will remain after translation.

Translation Rules for Adding Strings

To create a translation rule that adds a string:

1. Enter a descriptive name for this translation in the ID field.
2. Enter the string you want to add in the add string field.
3. Enter the index of the string insertion in the add-index parameter. If you want to add a string at the end of an number, enter a dollar-sign $ in the add index field.

Translation Rules for Replacing Strings

To create a translation rule that replaces a string:

> **Note:**
>
> A string replacement involves deleting a string followed by adding a string in the removed string’s place. The index is not used when replacing a string.

1. Enter a descriptive name for this translation in the ID field.
2. Enter the string you want to delete in the delete string field.
3. Enter the string you want to add in the add string field.

Session Translation

A session translation defines how translation rules are applied to calling and called numbers. Multiple translation rules can be referenced and applied using this element, which groups rules together and allows them to be referenced by one identifier.

There are two parameters in the session translation element. The rules calling parameter lists the translation rules to be applied to the calling number. The rules called parameter lists of translation rules to be applied to the called number.

The Oracle USM applies the translation rules in the order in which they are entered. They are applied cumulatively. For example, if this field is configured with a value of rule1 rule2 rule3, rule1 will be applied to the original number first, rule2 second, and rule3 last.

To configure the session translation element:

1. Enter a descriptive name for this session translation in the ID field.
2. Enter the IDs of existing translation rules in the rules calling parameter. Multiple rules can be entered in this field. The order you enter them in is the order in which they are applied.
3. Enter the IDs of existing translation rules in the rules called parameter. Multiple rules can be entered in this field. The order you enter them in is the order in which they are applied.

Applying Session Translations

Session translations can be applied to both session agents and realms. Both session agents and realms contain the two parameters that denote incoming and outgoing call legs—in translation ID and out translation ID. These two fields are populated with session translation element IDs.

If none of these fields are populated, no number translation will take place and the original address will remain unchanged as it traverses the Oracle USM. Further, any session translation applied to a session agent takes precedence over one applied to a realm.

Session Agent

To configure number translation for a session agent:

1. In the session agent element, set the in translation ID and/or the out translation ID to the appropriate ID you configured in the session translation element. There can be only one entry in each of these fields.

Realm

To configure number translation for a realm:

1. In the realm configuration element, set the in translation ID and/or the out translation ID to the appropriate ID you configured in the session translation element. There can be only one entry in each of these fields.

Number Translation Configuration

To create a translation rule:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **translation-rules** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# translation-rules
   ORACLE(translation-rules)#
   ```

   From this point, you can configure translation rules parameters. To view all translation rules parameters, enter a **?** at the system prompt. The following is an example of what a translation rule configuration might look like. Parameters not described in this section are omitted below.

   ```
   translation-rules
           id                          addplus1
           type                        add
           add-string                  +1
           add-index                   0
           delete-string
           delete-index                0
   ```

Translation Rules

Set the following parameters to configure a translation rule:

1. **ID**—Set a descriptive ID name for this translation rule.
2. **type**—Set the type of translation rule you want to configure. The default value is **none**. The valid values are:
   - **add**—Adds a character or string of characters to the address
   - **delete**—Deletes a character or string of characters from the address
   - **replace**—Replaces a character or string of characters within the address
   - **none**—Translation rule is disabled
3. **add-string**—Enter the string to be added during address translation to the original address. The value in this field should always be a real value; i.e., this field should not be populated with at-signs (@) or dollar-signs ($). The default value is a blank string.
4. **add-index**—Enter the position, 0 being the left most position, where you want to add the string defined in the **add-string** parameter. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
5. **delete-string**—Enter the string to be deleted from the original address during address translation. Unspecified characters are denoted by the at-sign symbol (@).

The @ character only works if the type parameter is set to delete. This parameter supports wildcard characters or digits only. For example, valid entries are: delete-string=@@@@@, or delete-string=123456. An invalid entry is delete-string=123@@@. When the type is set to replace, this value is used in conjunction with the add-string value. The value specified in the delete-string field is deleted and the value specified in the add-string field is inserted. If no value is specified in the delete-string parameter and the type field is set to replace, then nothing will be inserted into the address. The default value is a blank string.

6. **delete-index**—Enter the position, 0 being the left most spot, where you want to delete the string defined in the **delete-string** parameter. This parameter is only used if the **delete-string** parameter is set to one or more at-signs. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999

Session Translation

To configure session translations:

1. Exit out of the translation rules element and enter the session translation element.

```
ORACLE(translation-rules)# exit
ORACLE(session-router)# session-translation
ORACLE(session-translation)#
```

   From this point, you can configure the session translation element. To view all session translation parameters, enter a **?** at the system prompt. The following is an example of what a session translation configuration might look like:

```
session-translation
        id                              lrules-out
        rules-calling                   rule1 rule2 rule3
        rules-called                    addplus1
```

2. **ID**—Set a descriptive ID name for this session translation.
3. **rules-calling**—Enter the rules calling in the order in which they should be applied. Multiple rules should be included in quotes and separated by spaces.

```
ORACLE(session-translation)# rules-calling "rule1 rule2 rule3"
```

4. **rules-called**—Enter the rules called in the order in which they should be applied. Multiple rules should be included in quotes and separated by spaces.

Number Translation Application

To complete your number translation configuration, you must enter into a **realm-config** or **session-agent** element and assign **session-translations** there.

1. Navigate to the chosen element.
   - To move from the session-translation element to the session-agent element, exit out of the session translation element and enter the session agent element.

```
ORACLE(session-translation)# exit
ORACLE(session-router)# session-agent
ORACLE(session-agent)#
```

- To move from the session-translation element to the realm-config element, exit from the session translation element to the configuration path.

  ```
  ORACLE(session-translation)# exit
  ORACLE(session-router)# exit
  ```

  Navigate to the ream-config element located in the media-manager path.

  ```
  ORACLE(configure)# media-manager
  ORACLE(media-manager)# realm-config
  ORACLE(realm-config)#
  ```

2. In both **realm-config** or **session agent** elements, you must specify an **in-translationid** and/or an **out-translationid** in order to configure the number translation.

   ```
   session-agent
           in-translationid
           out-translationid               lrules-out
   realm-config
           in-translationid
           out-translationid               lrules-out
   ```

   Set the following parameters to configure a translation rule:

   - **in-translationid**—Enter the configured session translation that you want to affect the incoming traffic in this parameter.
   - **out-translationid**—Enter the configured session translation that you want to affect the outgoing traffic in this parameter.

Other Translations

FQDN Mapping

The Oracle USM maps FQDNs that appear in certain headers of incoming SIP messages to the IPv4 address that the Oracle USM inserts in outgoing SIP contact headers. The mapped FQDNs are restored in the SIP headers in messages that are sent back to the originator.

This feature is useful to carriers that use IPv4 addresses in the SIP From address to create trunk groups in a PSX for routing purposes. When the carrier's peer uses FQDNs, the carrier is forced to create trunk groups for each possible FQDN that it might receive from a given peer. Similarly, this can apply to SIP Contact and P-asserted-identity headers.

13

Admission Control and QoS

This chapter describes how to configure the Oracle USM for call admission control and Quality of Service (QoS) monitoring. Call admission control lets you manage call traffic based on several different policies. It is aimed at managing call admission rates in the network, enabling you to maintain suitable QoS levels. A new call is admitted only if it meets the requirements

QoS reporting provides you with real-time evaluation of network and route performance. It lets you contrast internal domain and external domain performance and facilitates SLA verification and traffic engineering.

About Call Admission Control

The Oracle USM provides call admission control capabilities based on the following policies:

- Bandwidth (single and multi-level policies)
- Session capacity
- Session rate (sustained and burst)

> **Note:**
>
> In order to provide admission control for networks to which the Oracle USM is not directly connected, you need to define multiple realms per network interface.

Bandwidth-Based Admission Control

The Oracle USM is a policy enforcement point for bandwidth-based call admission control. Sessions are admitted or rejected based on bandwidth policies, configured on the Oracle USM for each realm.

To manage bandwidth consumption of a network's overall capacity, you can configure aggregate bandwidth policies for each realm.

As the Oracle USM processes call requests to and from a particular realm, the bandwidth consumed for the call is decremented from the bandwidth pool for that realm. The Oracle USM determines the required bandwidth from the SDP/H.245 information for SIP and from the OLC sent in the SETUP message for H.323. Any request that would cause the bandwidth constraint to be exceeded is rejected with a SIP 503 Service Unavailable or an H.323 Release Complete.

For example, if an incoming SIP message requests PCMU for a payload/encoding name, a zero (0) payload type, and an 8000 cycle clock rate, the Oracle USM must determine how much bandwidth is needed.

To accomplish this task, the system checks the media profile values and reserves the bandwidth required for flows. If the required bandwidth for the new flow exceeds the available bandwidth at the time of the request, the Oracle USM rejects the session.

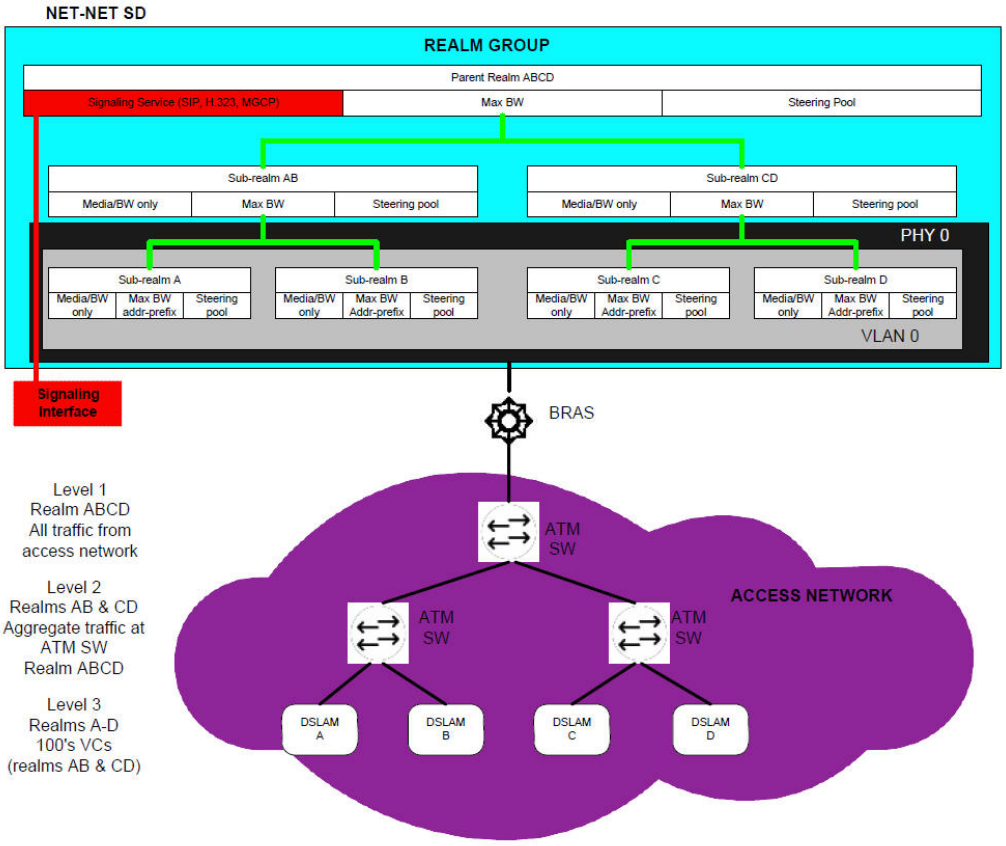
With these mechanisms, the Oracle USM provides bandwidth-based admission control.

Multi-Level Bandwidth Policy Nesting

Multi-level nesting of bandwidth policy enforcement addresses the following issues:

- Bandwidth over-subscription: access or transit transport networks are aggregated and/or oversubscribed. For example, digital subscriber lines (DSL), Frame Relay (FR), and Asynchronous Transfer Mode (ATM). Admission control policies must reflect access network topology.
- Bandwidth partitioning for multiple services: access or transit bandwidth is partitioned among multiple service profiles (SIP, for example) in the same customer network.
- Multi-site VPN environments: admission control must be applied at the site level as well as the VPN level.

The following example illustrates different scenarios; in each there are two or more levels of admission control required. Nested admission control is best depicted by the DSL broadband example.



In DSL access networks, ATM network bandwidth is typically oversubscribed at rates up to 400/1. At Level 3 (above), hundreds of users virtual circuits (VCs) are aggregated to a smaller set of virtual paths (VPs) at each DSLAM. At Level 2, many virtual paths are aggregated at the first ATM switch. Finally, at Level 1, all traffic from all subscribers in the access network is aggregated at the BRAS. Each level of aggregation is oversubscribed, creating the need to perform admission control at each level.

From a Oracle USM perspective, multiple tiers of realms are supported, each with its unique bandwidth policy. Only the lowest order realm (Level 3) requires an address prefix (that

assigned to the DSLAM) that must be used by the Oracle USM to determine in which realm a user resides. When a call request to or from a particular user is received, the Oracle USM checks each realm in the path to determine whether sufficient bandwidth is available to place the call.

Session Capacity- and Rate-based Admission Control

A session agent defines a signaling endpoint. It is a next hop signaling entity that can be configured to apply traffic shaping attributes. You can define concurrent session capacity and rate attributes for each session agent.

You can configure a set of attributes and constraints for each session agent to support session access control. In this configuration, the Oracle USM only accepts requests from configured session agents. And you can set up session admission control so that the Oracle USM limits the number of concurrent inbound and outbound sessions for any known service element.

The Oracle USM denies a call request to any destination that has exceeded its configured policies for session capacity and session rate. The Oracle USM might reject the call request back to the originator. If multiple destinations are available, the Oracle USM will check current capacity and rate for each destination and attempt to route the call only to destinations whose policy limits have not been reached.

You assign a media profile to a session agent and indicate whether the transport protocol is SIP or H.323. If the protocol is H.323, you need to indicate whether the session agent is a gateway or a gatekeeper.

Constraints for Proxy Mode

The Oracle USM applies session router and session agent constraints when it is in proxy (transaction or stateless) mode if you enable the ACLI **constraints** parameter for a session agent. However, the Oracle USM does not track SIP sessions when in transaction mode, so the following session-specific constraints are not applied:

- max-sessions
- max-inbound-sessions
- max-outbound-sessions
- min-seizures
- min-asr

Constraints the Oracle USM applies are:

- max-burst-rate
- max-inbound-burst-rate
- max-outbound-burst-rate
- max-sustain-rate
- max-inbound-sustain-rate
- max-outbound-sustain-rate

In order to set the desired time windows for computing burst rates and sustain rates, you also need to configure these parameters in the session agent configuration: **burst-rate-window** and **sustain-rate-window**. You can also set the **time-to-resume** and in-service-period parameters to

control how long to wait before bringing a session agent back into service after its constraints are no longer exceeded.

CAC Policing and Marking for non-Audio non-Video Media

The Oracle USM supports non-AVT (audio-visual transport) media profile and media policy configurations.

In previous releases, the Oracle USM only policed media based on average rate limits configured in media profiles, but these are only applied to AVT. And if there are not required bandwidth or average rate limit values set for the media profile, CAC and policing functions are not applied to media—even if the SDP specifies appropriate bandwidth values. Likewise, ToS markings are not applied for non-AVT media, but only for SIP, H.323, and AVT media types.

With this feature addition, you can now enable your Oracle USM to handle non-AVT media types like image and text, and use application and data type for policing purposes. Bandwidth CAC support has also been added for non-AVT media types, as has support for the application specific (AS) bandwidth modifier (b=AS:<value>) in the SDP with specification of a defined amount of headroom for that value.

Bandwidth CAC Fallback Based on ICMP Failure

For networks where backup links (operating in active-standby mode) from CE-routers to the MPLS backbone are provisioned with less bandwidth than the primary links, the Oracle USM can:

- Detect remote link failures
- Trigger bandwidth updates at the realm level when using backup links
- Detect remote link failback to primary

To do so, the Oracle USM monitors the primary link status using ICMP echo requests (or pings). It issues the pings at regular intervals, forming a heartbeat mechanism. The CE-router can respond to these pings on the primary link, which is represented by the WAN IP address. When this link fails over, the backup link assumes the same WAN IP address but is not responsive to the pings. This way, the Oracle USM determines failover when the ICMP ping fails.

When there is an ICMP ping failure, the Oracle USM adjusts the realm's available bandwidth pool from its maximum bandwidth setting to its fallback setting. If the fallback amount is less than the maximum amount, it is possible for the Oracle USM to start rejecting calls. It does so until enough calls are released to free adequate bandwidth to stay under the fallback limit and still accept calls.

Bandwidth CAC Fallback Based on ICMP Failure Configuration

You can set up ICMP heartbeats and fallback bandwidth pools in the realm configuration. Leaving the **icmp-detect-multiplier**, **icmp-advertisement-interval**, or **icmp-target-ip** parameters blank or set to zero turns the feature off.

To enable bandwidth CAC fallback based on ICMP failure:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **icmp-detect-multiplier**—Enter the multiplier you want to use when determining how long to send ICMP pings before considering a target unreachable. This number multiplied by the time you set for the **icmp-advertisement-interval** determines the length of time. For example, if you set this parameter to 10 and the advertisement interval to 20, the Oracle USM will send ICMP pings for 200 seconds before declaring the target unreachable.
5. **icmp-advertisement-interval**—Enter the time in seconds between ICMP pings the Oracle USM sends to the target. The default is 0.
6. **icmp-target-ip**—Enter the IP address to which the Oracle USM should send the ICMP pings so that it can detect when they fail and it needs to switch to the fallback bandwidth for the realm. There is no default.
7. **fallback-bandwidth**—Enter the amount of bandwidth you want available once the Oracle USM has determined that the target is unreachable.

   If the fallback amount is less than the **max-bandwidth** value, the Oracle USM might start to reject calls. It does so until enough calls are released to free adequate bandwidth to stay under the fallback limit and still accept calls.
8. Save and activate your configuration.

Bandwidth CAC for Aggregate Emergency Sessions

You can configure the maximum amount of bandwidth on your Oracle USM you want used specifically for priority (emergency) calls in the realm configuration's **max-priority-bandwidth** parameter. You set this limit on a per-realm basis, and the limit is enforced for nested realms. Setting a bandwidth limit specifically for priority calls allows the Oracle USM to reject calls exceeding the threshold, and also to accept calls that exceed the bandwidth limit for non-priority calls (set in the **max-bandwidth** parameter).

The bandwidth limit for emergency calls operates in conjunction with the bandwidth limits you can set for all other types of calls. When an emergency call comes in, the Oracle USM checks the non-priority bandwidth limit. If bandwidth is sufficient, the call goes through and the Oracle USM decrements the bandwidth used from the pool of the amount available.

However, if a priority call exceeds the **max-bandwidth** setting, the Oracle USM checks the **max-priority-bandwidth** parameter. If is it within the limit for priority calls, the system allows the call and decrements the amount of used bandwidth from what is available.

When there is not enough bandwidth in either the priority or non-priority pool, the Oracle USM rejects the call with the corresponding error code and reason phrase.

Any bandwidth subtracted from either pool during a session is returned to that pool as soon as the session ends.

Bandwidth CAC for Aggregate Emergency Sessions Configuration

You configure bandwidth CAC for priority calls on a per-realm basis.

> **Note:**
>
> This parameter honors the hierarchy of nested realms if you have them configured.

To enable bandwidth CAC for aggregate emergency sessions:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. If you are adding this feature to a pre-existing realm configuration, you will need to select and edit your realm.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **max-priority-bandwidth**Enter the amount of bandwidth you want to want to use for priority (emergency) calls. The system first checks the **max-bandwidth** parameter, and allows the call if the value you set for priority calls is sufficient. If there is not enough priority and non-priority bandwidth allotted for an incoming call, the Oracle USM rejects it.

   This parameter defaults to 0. You can enter any value between 0 and 999999999.

5. Save and activate your configuration.

Admission Control for Session Agents

This section explains how to configure session agents for admission control.

Session Agents Admission Control Configuration

To use admission control based on session rate, you need to configure session agent session rate constraints.

To configure session rates:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Enable session agent constraints and then configure the parameters related to session capacity or session rate to set admission control.

constraints—Enable this parameter. From here you can either configure admission control based on session capacity, session rates, or both. The default value is enabled. The valid values are:

- enabled | disabled

5. **max-sessions**—Set the maximum number of sessions (inbound and outbound) allowed by the session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295
6. **max-inbound-sessions**—Enter the maximum number of inbound sessions allowed from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
7. **max-outbound-sessions**—Enter the maximum number of concurrent outbound sessions (outbound from the Oracle USM) that are allowed from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295

   > **Note:**
   >
   > The number you enter here cannot be larger than the number you entered for max-sessions.

8. **max-burst-rate**—Enter a number to set how many SIP session invitations or H.323 SETUPs this session agent can send or receive (per second) within the configured burst rate window value. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—4294967295

     For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.

     For example, if you enter a value of 50 here and a value of 60 (seconds) for the burst rate window constraint, no more than 300 session invitations can arrive at or leave from the session agent in that 60 second time frame (window). Within that 60-second window, any sessions over the limit of 300 are rejected.
9. **max-inbound-burst-rate**—Enter the maximum burst rate (number of session invitations per second) for inbound sessions from this session agent. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
10. **max-outbound-burst-rate**—Enter the maximum burst rate (number of session invitations per second) for outbound sessions to this session agent.The default value is zero (**0**). The valid range is:
    - Minimum—0

- Maximum—999999999

11. **max-sustain-rate**—Enter a number to set the maximum rate of session invitations (per second) this session agent can send or receive within the current window. The default value is zero (**0**). The valid range is:
    - Minimum—zero (0)
    - Maximum—4294967295

    The number you enter here must be larger than the number you enter for **max-burst-rate**.

    For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.

    For example, if you enter a value of 50 here and a value of 36 (seconds) for the sustain rate window constraint, no more than 1800 session invitations can arrive at or leave from the session agent in any given 36 second time frame (window). Within that 36 second window, sessions over the 1800 limit are rejected.
12. **max-inbound-sustain-rate**—Enter the maximum sustain rate (of session invitations allowed within the current window) of inbound sessions from this session agent. This value should be larger than the **max-inbound-burst-rate** value. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
13. **max-outbound-sustain-rate**—Enter the maximum sustain rate (of session invitations allowed within the current window) of outbound sessions to this session agent. This value should be larger than the **max-outbound-burst-rate** value. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
14. **burst-rate-window**—Enter a number to set the burst window period (in seconds) that is used to measure the burst rate. The term window refers to the period of time over which the burst rate is computed. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—4294967295
15. **sustain-rate-window**—Enter a number to set the sustained window period (in seconds) that is used to measure the sustained rate. The default value is zero (**0**), which disables the functionality. The valid range is:
    - Minimum—10
    - Maximum—4294967295

    The value you set here must be higher than or equal to the value you set for the burst rate window.

> **Note:**
>
> If you are going to use this parameter, you must set it to a minimum value of 10.

The following example shows session agent constraints that are enabled and the session capacity parameters have been configured. Other session agent parameters have been omitted for brevity.

```
session-agent
constraints                    enabled
max-sessions                   355
max-inbound-sessions          355
max-outbound-sessions          355
```

The following example shows session agent constraints are enabled and the session rate parameters have been configured. Other session agent parameters have been omitted for brevity.

```
session-agent
max-burst-rate                 0
max-inbound-burst-rate         10
max-outbound-burst-rate        1
max-sustain-rate               3000
max-inbound-sustain-rate       0
max-outbound-sustain-rate      0
burst-rate-window              0
sustain-rate-window            0
```

Realm Bandwidth Configuration

To configure admission control based on bandwidth, you set the max and min bandwidth parameters in the realm configuration.

To configure realm bandwidth:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Configure the maximum bandwidth.

   max-bandwidth—Enter a number that sets the maximum bandwidth for dynamic flows to/from the realm in kilobits (Kbps) per second. The default value is zero (0). The valid range is:

   - Minimum—0
   - Maximum—4294967295

     The following example shows the maximum bandwidth for the realm has been configured. All other realm parameters have been omitted for brevity.

     ```
     realm-config
     max-bandwidth                  64000
     ```

SIP Admission Control Configuration

You can configure the registered endpoint to accept and process requests from SIP realms. If a request does not meet the criteria of the option you choose here, it is rejected with a 403 (Forbidden) response.

To configure admission control:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **sip-ports** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(sip-interface)# sip-port
   ORACLE(sip-port)#
   ```

5. Set the criteria for admission control.

   allow-anonymous—Enter the anonymous connection mode you want applied when SIP requests are processed. The default value is all.

   The following are valid values:

   - **all**—No ACL is applied and all anonymous connections are allowed.
   - **agents-only**—Only requests from configured session agents are processed. The Oracle USM responds to all other requests with a forbidden response.
   - **realm-prefix**—Only requests from session agents and addresses matching the realm's address prefix are processed. All other requests are rejected with a 403 (Forbidden) response.
   - **registered**—Only requests from session agents and registered endpoints are processed. REGISTER allowed from any endpoint.
   - **registered-prefix**—Only requests from session agent and registered endpoint addresses that match the realm's realm prefix are processed.

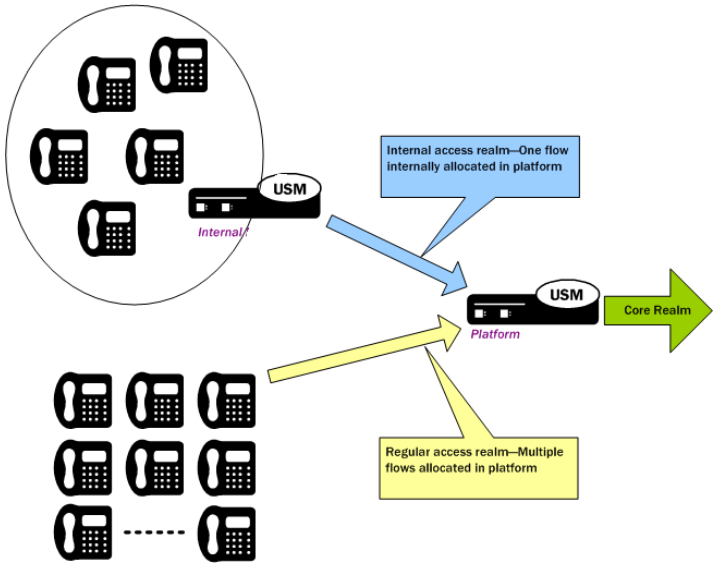
     The following example shows the **allow-anonymous** parameter that has been configured to allow only requests from session agents and registered endpoints. All other session agent parameters following the **allow-anonymous** parameters are omitted for brevity.

     ```
     sip-port
                    address
                    port                          5060
                    transport-protocol            UDP
                    allow-anonymous               registered
     ```

Session Agent Minimum Reserved Bandwidth

You can assign session agents minimum bandwidth, applicable in accessOracle USM deployments. Assigning a session agent minimum bandwidth can prevent overloading other network devices—such as another Oracle USM configured as a session agent. Doing so assures signaling bandwidth and availability to the endpoints behind this Oracle USM. This feature is only available on the Acme Packet 3820 and Acme Packet 4500.

In the following diagram, the internal Oracle USM is configured as a session agent on the platform Oracle USM (which conveys traffic to the core realm). Setting up bandwidth reservation allows for the creation of only one allocated flow, and secures bandwidth for all the SIP clients behind the internal Oracle USM. Contrast this scenario with the one where the platform Oracle USM must allocate multiple flows for many SIP clients.



When you configure minimum reserved bandwidth for session agent to a non-zero value, the Oracle USM allocates a separate pipe for per session agent. This is achieved by setting up an access control configuration in a specific way, instructing the Oracle USM to use a minimum number of transmission timeslots the individual pipe is guaranteed to receive.

This feature works across both signaling services: SIP and H.323. No more than 4000 session pipes are supported.

Session Agent Minimum Reserved Bandwidth Configuration

For this feature to work, you must set up an access control configuration as per below.

To configure minimum reserved bandwidth for session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **access-control** and press Enter.

   ```
   ORACLE(session-router)# access-control
   ORACLE(access-control)#
   ```

   If you are adding this feature to an existing configuration, you will need to select the configuration you want to edit.

4. **realm-id**—Enter the name of a valid realm.
5. **application-protocol**—Enter a valid application protocol. There is no default for this parameter, and valid values are: **SIP** or **H.323** .
6. **access**—Set this parameter to **permit** (default).
7. **trust-level**—Set this parameter to **high**, changing it from the default (**none**).
8. **minimum-reserved-bandwidth**—Enter the minimum reserved bandwidth you want for the session agent, and that will trigger the creation of a separate pipe for it. Only a non-zero value will allow the feature to work properly, along with the other required values set out in these instructions. The default is 0, and the maximum is 0xffffffff (or 4294967295).
9. Save and activate your configuration.

Aggregate Session Constraints for SIP

You can set a full suite of session constraints and then apply them to a SIP interface. The session constraints configuration contains many of the same parameters as the session agent, so you can configure a group of constraints and then apply them to a SIP interface/

The SIP interface configuration's **constraint-name** parameter invokes the session constraint configuration you want to apply. Using the constraints you have set up, the Oracle USM checks and limits traffic according to those settings for the SIP interface. Of course, if you do not set up the session constraints or you do not apply them in the SIP interface, then that SIP interface will be unconstrained. If you apply a single session-constraint element to multiple SIP interfaces, each SIP interface will maintain its own copy of the session-constraint.

SIP interfaces now have two states: "In Service" and "Constraints Exceeded." When any one of the constraints is exceeded, the status of the SIP interface changes to Constraints Exceeded and remains in that state until the **time-to-resume** period ends. The session constraint timers that apply to the SIP interface are the time-to-resume, burst window, and sustain window.

Aggregate Session Constraints Configuration

This section shows you how to configure aggregate session constraints and then apply them to a SIP interface.

The session constraints configuration contains many of the same parameters as the session agent does; it also incorporates the changes to the session agent parameters that are described in this section.

To configure the session constraints:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-constraints** and press Enter.

   ```
   ORACLE(session-router)# session-constraints
   ```

4. **name**—Enter the name for this session constraints configuration; this is a unique identifier that you will use in the SIP interface when you want the session constraints applied there. This is a required parameter that has no default.
5. **state**—Enable this parameter to use these session constraints. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **max-sessions**—Enter the maximum sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
7. **max-outbound-sessions**—Enter the maximum outbound sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
8. **max-inbound-sessions**—Enter the maximum inbound sessions allowed for this constraint. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
9. **max-burst-rate**—Enter the maximum burst rate (invites per second) allowed for this constraint. This value should be the sum of the **max-inbound-burst-rate** and the **max-outbound-burst-rate**. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—999999999
10. **max-sustain-rate**—Enter the maximum rate of session invitations per second allowed within the current window for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

      For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.
11. **max-inbound-burst-rate**——Enter the maximum inbound burst rate (number of session invitations per second) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
12. **max-inbound-sustain-rate**—Enter the maximum inbound sustain rate (of session invitations allowed within the current window) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.

13. **max-outbound-burst-rate**——Enter the maximum outbound burst rate (number of session invitations per second) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
14. **max-outbound-sustain-rate**—Enter the maximum outbound sustain rate (of session invitations allowed within the current window) for this constraint. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

    For the sustained rate, the Oracle USM maintains a current and previous window size. The period of time over which the rate is calculated is always between one and two window sizes.
15. **time-to-resume**—Enter the number of seconds after which the SA (Session Agent) is put back in service (after the SA is taken OOS (Out Of Service) because it exceeded some constraint). The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
16. **ttr-no-response**—Enter the time delay in seconds to wait before the SA (Session Agent) is put back in service (after the SA is taken OOS (Out Of Service) because it did not respond to the Oracle USM). The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
17. **in-service-period**—Enter the time in seconds that elapses before an element (like a session agent) can return to active service after being placed in the standby state. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
18. **burst-rate-window**—Enter the time in seconds that you want to use to measure the burst rate; the window is the time over which the burst rate is calculated, and is used for the over all burst rate as well as the inbound and outbound burst rates. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
19. **sustain-rate-window**—Enter the time in seconds used to measure the sustained rate; the window is the time over which the sustained rate is calculated, and is used for the over all sustained rate as well as the inbound and outbound sustained rates. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999

Applying Session Constraints in a SIP Interfaces

In the SIP interface, there is a new parameter that allows you to use a set of session constraints for that interface; the parameter is called constraint-name.

To apply session constraints to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. **constraint-name**—Enter the name of the session constraints configuration that you want to apply to this SIP interface. There is no default for this parameter.
5. Save and activate your configuration.

Configuring CAC Policing and Marking for non-Audio non-Video Media

In the media profile and the media policy configurations, the following values have been added for the **media-type** parameter:

- **application | data | image | text**

For the media policy, these new values apply to ToS marking.

Support for the AS Bandwidth Modifier

Two new parameters have been added to the media profile configuration:

- **sdp-bandwidth**—Enable or disable the use of the AS modifier in the SDP if the **req-bandwidth** and **sdp-rate-limit-headroom** parameters are not set to valid values in a corresponding media profile. The default value is **disabled**. The valid values are:
  - enabled | disabled
- **sdp-rate-limit-headroom**—Specify the percentage of headroom to be added while using the AS bandwidth parameter while calculating the **average-rate-limit** (rate limit for the RTP flow). The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—100

The following conditions apply to the use and application of these two new parameters:

- If the amount of required bandwidth is not specified in the media profile (**req-bandwidth**) for the media type in the m= line of the SDP, then the value specified in the AS modifier is used. The Oracle USM only uses the AS value if you set the new **sdp-bandwidth** to enabled.
- If the average rate limit value for RTP flows is not specified in the media profile (**average-rate-limit**) for the media type in the m= line of the SDP, then the value specified in the AS

modifier is used. The system only uses the AS value if you set the new **sdp-bandwidth** to enabled. When calculating the average rate rate limit that it will use based on the AS modifier, the Oracle USM applies the percentage set in the **sdp-rate-limit-headroom** parameter.

- The Oracle USM uses the value specified in the AS modifier (if **sdp-bandwidth** is enabled, and **req-bandwidth** is set to 0) along with the **user-cac-bandwidth** value set in the realm configuration; this works the same way that the **req-bandwidth** parameter does.
- The system uses the value specified in the AS modifier (if **sdp-bandwidth** is enabled, and **req-bandwidth** is set to 0) along with the **max-bandwidth** value set in the realm configuration; this works the same way that the **req-bandwidth** parameter does.

Media Profile Configuration

To set any of the new media types in the media profile configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **media-type**—Enter the media type that you want to use for this media profile. The valid values are:

   - **audio** | **video** | **application** | **data** | **image** | **text**

5. Save and activate your configuration.

   To set any of the new media types in the media policy configuration:

6. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

7. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

8. Type **media-policy** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# media-policy
   ORACLE(media-policy)#
   ```

9. **media-type**—Enter the media type that you want to use for this media profile. The valid values are:

   - **audio** | **video** | **application** | **data** | **image** | **text**

10. Save and activate your configuration.

AS Modifier and Headroom Configuration

To enable AS modifier use and establish the percentage of headroom to use:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **sdp-bandwidth**—Enable this parameter to use the AS bandwidth modifier in the SDP. The default is **disabled**. Valid values are:

   - enabled | disabled

5. **sdp-rate-limit-headroom**—Specify the percentage of headroom to be added while using the AS bandwidth parameter while calculating the **average-rate-limit** (rate limit for the RTP flow). The default is **0**. The valid range is:

   - Minimum—0
   - Maximum—100

6. Save and activate your configuration.

Offerless Bandwidth CAC for SIP

For SIP sessions offerless INVITEs (i.e., INVITEs that have no SDP offer), the Oracle USM can reserved bandwidth and support the session if you set up applicable media profile associations in the global SIP configuration. Otherwise, the Oracle USM terminates these sessions.

You configure support for offerless bandwidth CAC by setting up your global SIP configuration with the options parameters set to **offerless-media-bw-profiles**. The option takes multiple media profile names as values to apply when treating offerless INVITEs. When such an INVITE arrives and your configuration supports this option, the Oracle USM checks and reserves bandwidth for the session. If there is insufficient bandwidth to reserve, the Oracle USM terminates the session. Otherwise, the actual SDP negotiation takes place unaffected while the Oracle USM forwards the offerless INVITE. Once the negotiation completes, the Oracle USM updates bandwidth reservation.

If the called party's actual bandwidth needs exceed available bandwidth, the Oracle USM must terminate the session, even if the session is ringing or answered. To minimize this occurrence as much as possible, you should consider all case scenarios when you select media profiles to use with the **offerless-media-bw-profiles** option.

Offerless Bandwidth CAC for SIP Configuration

To configure offerless bandwidth CAC for SIP:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-config** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(session-router)# sip-config
ORACLE(sip-config)#
```

4. **options——Your entry will look like this:**

```
ORACLE(sip-config)# options offerless-bw-media-profiles=PCMU,G729
```

You can you the plus sign (+) and the minus sign (-) to add and remove values from the options list.

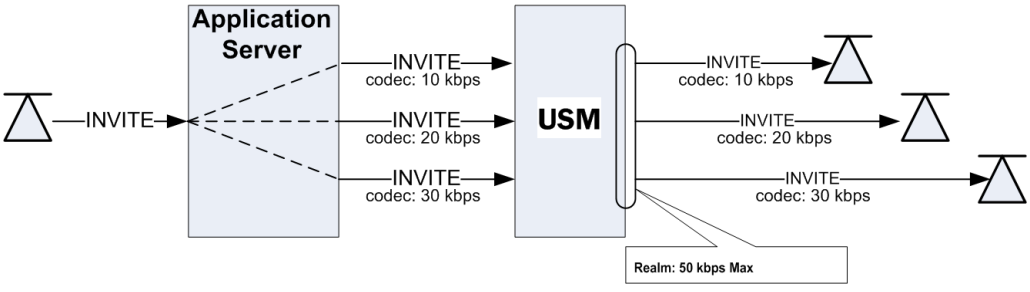
5. Type **done** and continue

Shared CAC for SIP Forked Calls

A forked call is one which has multiple INVITEs for the same call. For example, if an Application Server in the provider core network forks a call attempt, the application server sends several INVITEs for the same call toward the Oracle USM. Each INVITE is destined for a unique device that belongs to the same user. Ideally, that user will only answer one device. The Oracle USM treats each INVITE as a unique call request.

By default, each of the multiple INVITE forks are checked against CAC bandwidth limits, and thus they each consume bandwidth resources when they are received, even though only one of the forks will succeed in establishing a permanent session. Therefore, for many operators the CAC behavior of the SD is too restrictive and results in rejected call attempts which should have been allowed.

The following diagram shows a forked call scenario. The total bandwidth counted against the realm is 60 kbps. If the realm has a bandwidth ceiling of 50 kbps, one of the INVITEs will be rejected.



You can, however, enable the system to enforce CAC limits only once for SIP forked calls as long as the calls are identified as such, meaning that they will use the same bandwidth resources. The Oracle USM counts the forked call's most bandwidth-hungry codec at the time it arrives at the Oracle USM. In the above diagram, with shared bandwidth for forked calls enabled, the Oracle USM counts 30 kbps against the realm's total bandwidth after that INVITE arrives, even after the first two INVITES have passed into the final realm.

Bandwidth Sharing Scenarios

The following table summarizes how bandwidth would be shared given certain ingress and egress realms with this feature enabled. Realms A and C are call ingress realms.; realms B and D are egress realms. For the bandwidth to be shared, Call A and Call B must have the same forked Call-ID in the P-Multiring-Correlator header and be entering or exiting the Oracle USM on the same realm.

CALL A					
CALL B		Ingress Realm A	Egress Realm B	Ingress Realm C	Egress Realm D
	Ingress Realm A	bandwidth shared	N/A	bandwidth not shared	N/A
	Egress Realm B	N/A	bandwidth shared	N/A	bandwidth not shared
	Ingress Realm C	bandwidth not shared	N/A	bandwidth shared	N/A
	Egress Realm D	N/A	bandwidth not shared	N/A	bandwidth shared

Bandwidth Sharing Configuration

To enable bandwidth sharing of forked calls, set the **forked-cac-bw** parameter in the SIP profile configuration to **shared**. Although there are other parameters available in the SIP profile configuration, you only have to set the **name** and the **forked-cac-bw** values to use this feature.

After you set up the SIP profile, you apply it to a realm, SIP interface, or session agent.

Configuring a SIP Profile

The SIP profile is an element in the ACLI's **session-router** path, and you can configure multiple SIP profiles.

To configure a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter.

   ```
   ORACLE(session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **name**—Enter a name for this SIP profile configuration. This parameter is blank by default, and it is required. You will need the SIP profile's **name** when you want to apply this profile to a realm, SIP interface, or SIP session agent.

5. **forked-cac-bw**—Set this parameter to **shared** if you want forked sessions to share bandwidth resources, or set it to **per-session** if you want bandwidth to be counted for each session individually. There is no default for this parameter, and leaving it blank means:
   - For an ingress session agent without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle USM will reference the associated realm.
   - For an ingress realm without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle USM will reference the associated SIP interface.
   - For an ingress SIP interface without a SIP profile or with a SIP profile where the forked CAC mode is blank, the Oracle USM will not perform bandwidth sharing for forked calls.
6. Save your work.

Applying a SIP Profile

Once you have configured one or more SIP profiles, you can apply them to realms, SIP interfaces, and SIP session agents. As an example, this section shows you how to apply a SIP profile to a SIP interface. But the parameter name is the same in these configurations:

- realm-config
- sip-interface
- session-agent

To apply a SIP profile to a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **sip-profile**—Enter the name of SIP profile configuration that includes the forked-cac-bandwidth parameter configured.
5. Save your work.

RADIUS Accounting Support

VSA 171, Acme-Session-Forked-Call-Id, is part of the Oracle RADIUS dictionary. The VSA is a string value, and appears as the header-value without the header parameters from the P-Multiring-Correlator header for a session identified as part of a forked call.

Monitoring

Using the ACLI **show sipd forked** command, you can display the total number of forked sessions the Oracle USM received and the total number it rejected. The Oracle USM counts forked sessions when it receives a dialog-creating INVITE and is enabled to shared bandwidth. Further, it counts as forked all session with the P-Multiring-Correlator header.

```
ORACLE# show sipd forked
11:19:20-116
Forked Sessions               ---- Lifetime ----
                        Recent      Total  PerMax
Forked Sessions              0          0       0
Forked Sessions Rej          0          0       0
```

Conditional Bandwidth CAC for Media Release

The Oracle USM supports conditional call admission control (CAC) using the SIP profile configuration. With this feature enabled, you can allow the conditional admission of SIP calls that could potentially have their media released instead of risking the possible rejection of those calls due to internal bandwidth limits.

About Conditional Bandwidth CAC for Media Release

The Oracle USM performs bandwidth CAC for SIP per realm, for each Address of Record (AoR) or IP address. The system checks bandwidth limits based on the codecs listed in SDP. If a new SIP INVITE contains codecs in an SDP message that exceed bandwidth available for a given resource, the system rejects that INVITE. This check occurs both on the ingress and egress sides of a call, and both sides must have enough available resources to support the call for it to be admitted.

In the case of calls where media is released, the Oracle USM does not count bandwidth consumed by the call. However, this exemption is not given until the media is actually released —and media release conditions are unknown at the time SIP INVITE is admitted. This is because an INVITE received on one side of the Oracle USM is only media-released when that INVITE is routed back through the Oracle USM as a hairpin or other multi-system media release. So there has to be enough bandwidth for the initial INVITE; otherwise, and even if the INVITE is a candidate for media release, it will be rejected.

When there is a significant volume of such calls—ones that are candidates for media release, but cannot be admitted because of CAC limits—it becomes important to admit them so long as they truly end in media release. This feature thus allows admission of SIP calls that might otherwise be rejected because of bandwidth limitations when the far-end of the call causes media to be released.

Details and Conditions

This feature applies in a two system scenario. In order to track a call as a candidate for provisional media release, the access-side Oracle USM adds a Require: header with an option tag to the INVITE or UPDATE message on egress. The option tag is configurable in the sip config option. The default is com.acmepacket.cac .

The following sections describe when the SIP INVITE or SIP UPDATE are:

- initially received by the Oracle USM
- received by the second Oracle USM

INVITEs UPDATEs Initially Received By Oracle USM

When the Oracle USM first receives an INVITE or UPDATE message, it considers if it should be admitted provisionally or rejected outright due to CAC bandwidth constraints. If the INVITE or UPDATE is admitted provisionally, a Require: header is inserted on egress from the system.

The Oracle USM inserts the Require header on egress under these conditions:

- It receives an INVITE / UPDATE with no or a non-matching Require header.
- The **egress conditional cac admit** parameter in the SIP profile on the egress realm, SIP interface, session agent is set to enabled in the egress realm
- The request would otherwise be rejected because of current bandwidth CAC limits in the ingress OR egress realms
- The call is a candidate for media-release in the ingress realm

A call is considered a candidate for media-release when the ingress realm has any of these parameters set to disabled:

- **mm-in-realm**
- **mm-in-network**
- **mm-same-ip**
- **mm-in-system**

INVITEs UPDATEs Received by Second SBC

The second Oracle USM receives the INVITE or UPDATE with the newly inserted Require: header. Standard SIP convention indicates that if the UAS receiving the request does not know how to handle the Require header, the request should be rejected.

When the following three conditions are met, the INVITE is permitted into the system for processing:

- The **ingress conditional cac admit** in the SIP profile on the ingress realm, SIP interface, session agent parameter is set to enabled
- The **con-cac-tag** sip config option is configured to the same value as the received Require header's option tag
- The call is a candidate for media-release

The call is considered a candidate for media-release on the second system (indicated by the **ingress conditional cac admit** parameter is set to enabled) when either the ingress or egress realms have any of these parameters set to disabled:

- **mm-in-realm**
- **mm-in-network**
- **mm-same-ip**
- **mm-in-system**

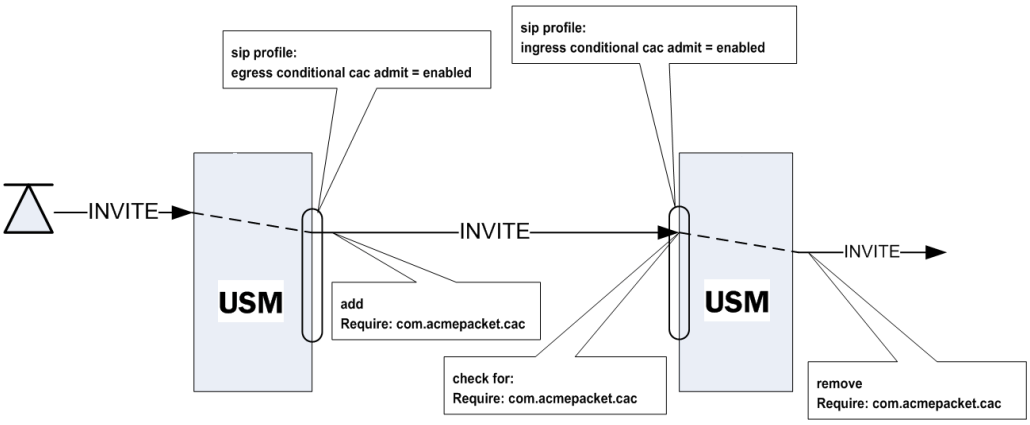
and the following parameter is set to enabled:

- msm-release

If the call, as received by the second system is not considered a candidate for release, the INVITE or UPDATE is failed with a 503 Insufficient Bandwidth message.

After the INVITE has been processed by the Oracle USM, the Require: header is removed upon egress from the system.

The following diagram shows the two-system scenario:



Conditional Admission with Per-user CAC

In the event that the per-user CAC feature is also being used, and per-user CAC bandwidth is exceeded, the Oracle USM also uses this option tag mechanism. However, if the per-user CAC implementation does count bandwidth regardless of media-release, then the Oracle USM will reject calls exceeding the per-user CAC limits when it receives them.

On the second system, when the per-user CAC feature is being used, the Oracle USM will perform the same option tag mechanism based on if the **ingress conditional cac admit** parameter is enabled.

Conditional Bandwidth CAC Configuration

You enable this feature by first configuring a SIP profile, and then applying the profile to any of these:

- realm
- SIP interface
- SIP session agent

SIP Profile Configuration

The SIP profile is an element in the ACLI's **session-router** path, and you can configure multiple SIP profiles. Though this configuration contains additional parameters, you do not have to use them for the conditional bandwidth CAC for media release.

To configure a SIP profile:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-profile** and press Enter.

   ```
   ORACLE(session-router)# sip-profile
   ORACLE(sip-profile)#
   ```

4. **name**—Enter a name for this SIP profile configuration. This parameter is blank by default, and it is required. You will need the SIP profile's **name** when you want to apply this profile to a realm, SIP interface, or SIP session agent.
5. **ingress-conditional-cac-admit**—Set this parameter to enabled to process an INVITE with a Require tag as received on an ingress interface. You can set this parameter to disabled if you do not want to use this feature on the ingress side. There is no default for this parameter.
6. **egress-conditional-cac-admit**—Set this parameter to enabled if you want to use conditional bandwidth CAC for media release for calls that are first received by this system. This results in option tags being inserted on the INVITE's egress if the conditional CAC conditions are met. You can set this parameter to disabled if you do not want to use this feature. There is no default for this parameter.
7. Save your work.

Applying a SIP Profile

Once you have configured one or more SIP profiles, you can apply them to realms, SIP interfaces, and SIP session agents. As an example, this section shows you how to apply a SIP profile to a SIP interface. But the parameter name is the same in these configurations:

- realm-config
- sip-interface
- session-agent

To apply a SIP profile to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
ORACLE(configure)#
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
ORACLE(session-router)#
```

3. Type **sip-interface** and press Enter.

```
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

4. **sip-profile**—Enter the name of SIP profile configuration you want to use for conditional bandwidth CAC for media release for this SIP interface. This value is blank by default, but it must be the value of the **name** parameter from a valid SIP profile.
5. Save your work.

Configuring Require Header Option Tag

You may change the Require: header's option tag from the default com.acmepacket.cac to one of your own choosing. Remember that both systems' option tags must match exactly.

To configure the Require: header's option tag:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter to access the **session-router** path.

```
ORACLE(configure)# session-router
```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ```

4. Use the ACLI **select** command so that you can work with the SIP configuration.

   ```
   ORACLE(sip-config)# select
   ```

5. **options**—Set the options parameter by typing +**options**, a Space, the option name **con-cac-tag=** your-new-tag, and then press Enter.

   ```
   ORACLE(sip-config)# options +con-cac-tag=com.test.cac
   ```

6. Save your work.

CAC Utilization Statistics via SNMP

The Oracle USM allows you to retrieve information on current session utilization and burst rate as a percentage of their configured maximums on per session-agent and/or realm basis. The Oracle USM uses the configured max-session and max-burst-rate settings in conjunction with a percentage formula to calculate this value. The system also uses an ACLI configuration setting to establish the threshold at which trap and trap clear messages are sent from the SNMP agent to the configured manager(s).

The user must load the MIB version associated with this software version on all pertinent SNMP managers to query these CAC utilization (occupancy) values and interpret the traps. In addition, the user must configure the threshold at which the system generates the CAC utilization trap. Note that the corresponding clear trap uses the same threshold setting, sending the clear trap when utilization falls below 90% of the threshold.

SNMP Get for CAC Utilization

Using a MIB browser, the user can query the current percentage utilization values for both max-session and max-burst-rate for any session-agent or realm. The calculations for these utilization levels are:

- Session utilization level = (current session count * 100) / max-sessions
- Burst rate utilization level = (current burst rate * 100) / max-burst-rate

The MIB objects associated with these statistics are parallel for session agent and realm and include a table to contain the objects, an object associating the objects containing the values with the applicable table, and objects containing the values themselves. These objects are listed below.

The MIB objects containing CAC utilization data for Session Agents are listed below.

The object establishing the statistics table for session agent CAC utilization follows:

```
--apSip Session Agent Connection Admission Control Stats Table
apSipSaCacStatsTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApSipSaCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "SIP Session Agent Connection Admission Control Stats Table."
        ::= { apSipMIBTabularObjects 5 }
```

The object establishing the session agent CAC utilization statistics objects follows:

```
apSipSaCacStatsEntry OBJECT-TYPE
        SYNTAX          ApSipSaCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Connection Admission Control Statistics."
        AUGMENTS { apSipSessionAgentStatsEntry }
     ::= { apSipSaCacStatsTable 1 }
```

The session agent CAC utilization statistics values include:

```
ApSipSaCacStatsEntry ::= SEQUENCE {
    apSipSaCacSessionUtilLevel                Gauge32,
    apSipSaCacBurstRateUtilLevel              Gauge32
}
```

The above objects, specifying the CAC utilization value for sessions and burst rate utilization for session agents include:

```
apSipSaCacSessionUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current session utilization level."
    ::= { apSipSaCacStatsEntry 1 }


apSipSaCacBurstRateUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current burst rate utilization level."
    ::= { apSipSaCacStatsEntry 2 }
```

The MIB objects containing CAC utilization data for Realms are listed below.

The object establishing the statistics table for realm CAC utilization follows:

```
--apSig Realm Connection Admission Control Stats Table
apSigRealmCacStatsTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF ApSigRealmCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Realm Connection Admission Control Stats Table."
        ::= { apSipMIBTabularObjects 6 }
```

The object establishing the realm CAC utilization statistics objects follows:

```
apSigRealmCacStatsEntry OBJECT-TYPE
        SYNTAX          ApSigRealmCacStatsEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
            "Connection Admission Control Statistics."
        AUGMENTS { apSigRealmStatsEntry }
     ::= { apSigRealmCacStatsTable 1 }
```

The session agent CAC utilization statistics values include:

```
ApSigRealmCacStatsEntry ::= SEQUENCE {
    apSigRealmCacSessionUtilLevel              Gauge32,
    apSigRealmCacBurstRateUtilLevel            Gauge32
}
```

The above objects, specifying the CAC utilization value for sessions and burst rate utilization for realms include:

```
apSigRealmCacSessionUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current session utilization level."
    ::= { apSigRealmCacStatsEntry 1 }

apSigRealmCacBurstRateUtilLevel        OBJECT-TYPE
    SYNTAX        Gauge32
    UNITS        "percentage"
    MAX-ACCESS     read-only
    STATUS        current
    DESCRIPTION
        "Current burst rate utilization level."
    ::= { apSigRealmCacStatsEntry 2 }
```

CAC Utilization Traps

The Oracle USM can issue a trap when either the value of max-session or CAC burst rate exceeds a configured value. The system only sends one trap when the threshold is exceeded. When the value falls back under 90% of this threshold, the Oracle USM sends a clear trap.

You configure the value that triggers these traps as a percentage of the max-session and max-burst-rate settings configured for the applicable session agent and/or realm. The system uses the same setting to specify when to send both the sessions and burst rate traps. The name of this parameter is the **cac-trap-threshold**.

For realms, you configure a **session-constraint** element with the **cac-trap-threshold** setting and apply that session constraint to the realm. For a session agent however, you configure the **cac-trap-threshold** directly within the session agent's configuration.

The syntax for the command is the same within session constraint and session agent configurations.

cac-trap-threshold[0-99]

You must express the value as a number less than 100. There is no default setting; the system does not generate a trap if you have not configured this setting.

The apSipCACUtilAlertTrap identifies the threshold exceeded on a per-element and per-value (session count or burst rate) for each trap, including:

- apSipSaCacSessionUtilLevel
- apSipSaCacBurstRateUtilLevel
- apSipRealmCacSessionUtilLevel
- apSipRealmCacBurstRateUtilLevel

CAC utilization threshold trap on a session agent configuration

The CAC utilization threshold causes the system to generate a trap when session count or CAC max-burst-rate exceeds the configured percentage value of these values maximums. This setting is available within a session agent's configuration.
To configure the CAC trap threshold on a session agent, follow the procedure below.

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813

selection: 1
ORACLE(session-agent)#
```

3. **cac-trap-threshold**—Set the threshold when reached, expressed as a percentage of **max-sessions**, when the CAC trap is sent.
4. Type **done** to save your configuration.

Configuring the CAC Utilization Thresholds - **realm**

To configure the CAC trap threshold on a realm or sip interface, create a session constraint object and apply it to your realm, as shown below.

1. Use the following sequence to navigate to session constraint elements.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-constraints
ORACLE(session-constraints)#
```

2. Select or create the desired session constraint element.

```
ORACLE(session-constraints)#name trap-at-90-percent
```

3. Configure the desired value for **cac-trap-threshold** expressed as a percentage value, such as 90%, as follows.

```
ORACLE(session-constraints)#cac-trap-threshold 90
```

4. Navigate to the realm-config to which you want to apply the session constraint.

```
ORACLE(realm-config)#session-constraint trap-at-90-percent
```

5. Execute the **done** and **exit** commands.
6. Save and activate your configuration.

About QoS Reporting

This section describes the Oracle USM QoS reporting. QoS reporting provides you with real-time evaluation of network and route performance. It lets you contrast internal domain and external domain performance and facilitates SLA verification and traffic engineering. Oracle USM QoS reporting is a measurement tool that collects statistics on Voice over IP (VoIP) call flows for SIP and H.323. To provide information, the Oracle USM writes additional parameters to the Remote Authentication Dial-in User Service (RADIUS) call record. To provide information, the Oracle USM writes additional parameters to the Remote Authentication Dial-in User Service (RADIUS) call record and Historical Data Recording (HDR) records.

You can use QoS statistics for SLA customer reporting, fault isolation, SLA verification, and traffic analysis. The Oracle USM employs specialized hardware to inspect Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP) flows while maintaining wire-speed packet forwarding. QoS metrics are collected and reported on a per-session and per call-leg basis. These metrics are reported through real-time RADIUS records along with call accounting data.

Overview

When a conversation is established between two endpoints, two flows are present in each direction:

- RTP flow carries traffic between endpoints with a predictable packet arrival rate. The packets headers have sequence numbers that are used to determine whether packets are missing or lost.
- RTCP flow carries information about the RTP flow and keeps a different record. The RTCP packets contain timestamps based on Network Time Protocol (NTP).

QoS Statistics

Reported QoS data includes the following per-flow statistics:

- RTP and RTCP lost packets—Count of lost packets for both RTP and RTCP based on comparing the sequence numbers since the beginning of the call or the last context memory poll.
- RTP and RTCP average jitter—Incremental number of packets for both RTP and RTCP that have been used to generate the total and max jitter since the beginning of the call or the last context memory poll. The incremental accumulated jitter (in milliseconds) over all the packets received.
- RTP and RTCP maximum jitter—Maximum single jitter value (in milliseconds) for both RTP and RTCP from all the packets since the beginning of the call or the last context memory poll.
- RTCP average latency—Number of RTCP frames over which latency statistics have been accumulated and the incremental total of latency values reported since the beginning of the call or the last context memory poll.
- RTCP maximum latency—Highest latency value measured since the beginning of the call or the last context memory poll.
- RTP packet count
- RTP bytes sent and received

- RTCP lost packets—RTP lost packets reported in RTCP packets.
- ATP lost packets—Lost packets determined by monitoring RTP sequence numbers.
- R-Factor and MOS data—R-Factor and MOS data for the calling and called segments at the end of a session

RADIUS Support

All the QoS statistics go into the RADIUS CDR. If a RADIUS client is configured on the Oracle USM, any time a call occurs a record is generated and sent. Only Stop RADIUS records contain the QoS statistic information.

Only RADIUS Stop records contain QoS information. For non-QoS calls, the attributes appear in the record, but their values are always be zero (0). When you review the list of QoS VSAs, please note that “calling” in the attribute name means the information is sent by the calling party and called in the attribute name means the information is sent by the called party.

The following example shows a CDR that includes QoS data:

```
Wed Jun 13 18:26:42 2007
        Acct-Status-Type = Stop
        NAS-IP-Address = 127.0.0.100
        NAS-Port = 5060
        Acct-Session-Id = "SDgtu4401-c587a3aba59dcae68ec76cb5e2c6fe6f-v3000i1"
        Acme-Session-Ingress-CallId =
"8EDDDC21D3EC4A218FF41982146844310xac1ec85d"
        Acme-Session-Egress-CallId = "SDgtu4401-c587a3aba59dcae68ec76cb5e2c6fe6f-
v3000i1"
        Acme-Session-Protocol-Type = "SIP"
        Calling-Station-Id = ""9998776565" <sip:
9998776565@10.10.170.2:5060>;tag=2ed75b8317f"
        Called-Station-Id = "<sip:7143221099@10.10.170.2:5060>"
        Acct-Terminate-Cause = User-Request
        Acct-Session-Time = 7
        h323-setup-time = "18:24:36.966 UTC JUN 13 2007"
        h323-connect-time = "18:24:37.483 UTC JUN 13 2007"
        h323-disconnect-time = "18:24:44.818 UTC JUN 13 2007"
        h323-disconnect-cause = "1"
        Acme-Session-Egress-Realm = "peer"
        Acme-Session-Ingress-Realm = "core"
        Acme-FlowID_FS1_F = "localhost:65544"
        Acme-FlowType_FS1_F = "PCMA"
        Acme-Flow-In-Realm_FS1_F = "core"
        Acme-Flow-In-Src-Addr_FS1_F = 10.10.170.15
        Acme-Flow-In-Src-Port_FS1_F = 49156
        Acme-Flow-In-Dst-Addr_FS1_F = 10.10.170.2
        Acme-Flow-In-Dst-Port_FS1_F = 31008
        Acme-Flow-Out-Realm_FS1_F = "peer"
        Acme-Flow-Out-Src-Addr_FS1_F = 10.10.130.2
        Acme-Flow-Out-Src-Port_FS1_F = 21008
        Acme-Flow-Out-Dst-Addr_FS1_F = 10.10.130.15
        Acme-Flow-Out-Dst-Port_FS1_F = 5062
        Acme-Calling-RTCP-Packets-Lost_FS1 = 0
        Acme-Calling-RTCP-Avg-Jitter_FS1 = 15
        Acme-Calling-RTCP-Avg-Latency_FS1 = 0
        Acme-Calling-RTCP-MaxJitter_FS1 = 15
        Acme-Calling-RTCP-MaxLatency_FS1 = 0
        Acme-Calling-RTP-Packets-Lost_FS1 = 0
        Acme-Calling-RTP-Avg-Jitter_FS1 = 3
        Acme-Calling-RTP-MaxJitter_FS1 = 44
```

```
Acme-Calling-Octets_FS1 = 957
Acme-Calling-Packets_FS1 = 11
Acme-FlowID_FS1_R = "localhost:65545"
Acme-FlowType_FS1_R = "PCMA"
Acme-Flow-In-Realm_FS1_R = "peer"
Acme-Flow-In-Src-Addr_FS1_R = 10.10.130.15
Acme-Flow-In-Src-Port_FS1_R = 5062
Acme-Flow-In-Dst-Addr_FS1_R = 10.10.130.2
Acme-Flow-In-Dst-Port_FS1_R = 21008
Acme-Flow-Out-Realm_FS1_R = "core"
Acme-Flow-Out-Src-Addr_FS1_R = 10.10.170.2
Acme-Flow-Out-Src-Port_FS1_R = 31008
Acme-Flow-Out-Dst-Addr_FS1_R = 10.10.170.15
Acme-Flow-Out-Dst-Port_FS1_R = 49156
Acme-Called-RTCP-Packets-Lost_FS1 = 0
Acme-Called-RTCP-Avg-Jitter_FS1 = 13
Acme-Called-RTCP-Avg-Latency_FS1 = 0
Acme-Called-RTCP-MaxJitter_FS1 = 21
Acme-Called-RTCP-MaxLatency_FS1 = 0
Acme-Called-RTP-Packets-Lost_FS1 = 0
Acme-Called-RTP-Avg-Jitter_FS1 = 0
Acme-Called-RTP-MaxJitter_FS1 = 3
Acme-Called-Octets_FS1 = 77892
Acme-Called-Packets_FS1 = 361
Acme-Firmware-Version = "C5.0.0"
Acme-Local-Time-Zone = "Time Zone Not Set"
Acme-Post-Dial-Delay = 110
Acme-Primary-Routing-Number = "sip:7143221099@10.10.170.2:5060"
Acme-Ingress-Local-Addr = "10.10.170.2:5060"
Acme-Ingress-Remote-Addr = "10.10.170.15:5060"
Acme-Egress-Local-Addr = "10.10.130.2:5060"
Acme-Egress-Remote-Addr = "10.10.130.15:5060"
Acme-Session-Disposition = 3
Acme-Disconnect-Initiator = 2
Acme-Disconnect-Cause = 16
Acme-SIP-Status = 200
Acme-Egress-Final-Routing-Number = "sip:7143221099@10.10.130.15:5060"
Acme-CDR-Sequence-Number = 14
Client-IP-Address = 172.30.20.150
Acct-Unique-Session-Id = "0832b03cd3a290b3"
Timestamp = 1181773602
```

Configuring QoS

This section explains how to configure QoS. To generate QoS metrics, you need to enable QoS for the realm of the originating caller. The ingress realm determines whether QoS is turned on for a specific flow.

> **Note:**
>
> If you run with QoS turned on one side only and disabled on the other you lose the ability to measure latency through the use of RTCP timestamps.

QoS Configuration

To enable QoS:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **qos-enable**—Enable this parameter. The default value is **disabled**.

Network Management Controls

The Oracle USM supports network management controls for multimedia traffic specifically for static call gapping and 911 exemption handling. These controls limit the volume or rate of traffic for a specific set of dialed numbers or dialed number prefixes (destination codes).

In TDM networks, automatic call/code gapping was developed as part of the advanced intelligent network (AIN) to enable network element load shedding based on destination number (DN) in case of overload. However, since there are as yet no standards for call/code gapping for next generation multimedia networks, the Oracle USM provides statically-provisioned network management controls.

To enable network management controls on your Oracle USM, you set up the ACLI **net-management-control** configuration and then enable the application of those rules on a per-realm basis. Each network management control rule has a unique name, in addition to information about the destination (IP address, FQDN, or destination number or prefix), how to perform network management (control type), whether to reject or divert the call, the next hop for routing, and information about status/cause codes. Details about the content of control rules and how to set them appear in the instructions and examples section.

When a SIP INVITE or an H.323 Setup for a call arrives at the Oracle USM on an ingress realm where network management controls have been enabled, the Oracle USM takes the following steps:

- It searches the network management rules you have created (which are stored in tables on the Oracle USM) for a control rule that best matches the newly-received call.
- If it does not find a matching control rule, the Oracle USM allows the call to proceed normally.
- If it finds a matching control rule, then the Oracle USM treats the call according to the specifics of the rule and the treatment method that it designates.

Matching a Call to a Control Rule

The Oracle USM uses the call classification key (specified by the **destination-identifier** parameter) to match calls so that it can apply control rules. The call classification key specifies information about the destination, which can be an IP address, an FQDN, a destination (called) number, or destination prefix. You configure the classification key as part of the control rule.

Matching is performed from left to right, starting at the left-most character. A wildcard matches any digit.

The Oracle USM compares the following information from the SIP INVITE or H.323 Setup for matching:

- SIP INVITE—User part of the Request URI, or the host part of the Request URI
- H.323 Setup—Q.931 Called Party Number IE

With Release 6.0, the Oracle USM now normalizes the user-part of the Request-URI prior to performing any matching for NMC based on the dialed number. A departure from this feature's prior implementation, this normalization strips out any of the visual-separator characters.

Note that normalization occurs only for NMC look-up purposes, and it does not alter the actual Request-URI. For previous releases, NMC rule matching based on the dialed number fails when the dialed number has visual separators or additional parameters such as: rn, npdi, cic, postd, etc. If multiple rules match an incoming call, then the Oracle USM gives first priority to destination number or the destination prefix. Next, it tries to match based on the IP address, and finally it looks to the domain (lowest priority).

Specifically, the Oracle USM supports the following:

- The user-part can contain escaped sequences that the Oracle USM normalizes to their unescaped representation. For examples %23(358)555.1234567 would be normalized to #3585551234567.
- The Oracle USM parses the user-part of the Request-URI up to the first semicolon (;). For example, the user-part in tel:+358-555-1234567;postd=pp22 will be +358-555-12134567.

Call Handling Determination

There are three types of control rules from which you can choose; each is a different way for the Oracle USM to handle calls matching the classification key:

- Call gap rate—Controls the maximum sustained rate of calls that match the classification key.
  Using this type, the Oracle USM calculates the time since the last matching call. If that time is equal to or greater than the minimum time gap set in the control rule (i.e., it does not exceed the rate), then the call proceeds normally. If the call is less than the minimum time gap (i.e., it causes the call rate to be exceeded), then the Oracle USM either rejects or diverts the call.

  To keep the call rate below the control value, the Oracle USM ensures a minimum call gap time between the matching calls. For example, if the control value is 10 calls per second, the minimum call gap time would be 0.1 second. And if a matching call were to arrive within a tenth of a second since the last matching call, then the Oracle USM applies the treatment method.
- Call gap percentage—Controls the percentage of calls matching the classification key you set for the control rule.
  When using this control rule type, the Oracle USM applies the treatment method to the percentage of matching calls (that you set in the value parameter) out the total number of matching calls it receives. For example, if you set the value parameter for the control rule to 50 and use this control type, the Oracle USM applies the treatment method to every other call it receives (or 50% of the calls it receives) that matches the classification key.

  Note that the Oracle USM cannot maintain exact percentages for the control value at all times, especially at system start-up when the number of incoming calls is small.
- Priority—Exempts calls to a destination (like 911) from local network management controls such as:

  - Session agent constraints
  - Bandwidth constraints (such as per-realm bandwidth)
  - External policy servers (requests are made to the policy server; calls are admitted and processed regardless of the decision or reachability of the policy server)
  - Per-user call admission control
  - CPU constraints
    The Oracle USM will not bypass licensing constraints, however.

Treatment Methods

You can choose from two different treatment methods:

- Call rejection—The Oracle USM rejects the call.
  - For SIP, the Oracle USM sends a response messages with the status code of the control rule. This response message also includes a Reason header with the Q.850 cause code that you configure as part of the control rule; it contains only the Q.850 cause code, and there is no reason-text included. For example:

    ```
    Reason: Q.850; cause=63
    ```

  - For H.323, the Oracle USM sends a releaseComplete message with the Q.850 cause code (that you configure as part of the control rule) of the control rule as the Q.931 Cause IE.
- Call diversion—The Oracle USM routes the call to the location you specify in the control rule's next hop parameter.
  Except for this routing, the call proceeds as normal. Local treatments such as number translation apply to the call, as do local controls such as licensing. Note the following:
  - If the next hop is an FQDN, the Oracle USM performs DNS queries to resolve the next hop to an IP address so that it can route the call properly.
    DNS queries only apply to pure SIP or IWF calls that originate in H.323 and are interworked to SIP.
  - If the next hop is a session agent group, the Oracle USM selects a session agent from the group according to the selection strategy you set for the group. Then the Oracle USM uses the IP address of the selected session agent.

Priority Call Exemption from Policy Server Approval

The Oracle USM now identifies priority calls and provides expedited treatment for them, even is these calls use associated realms for which there is an associated policy server handling bandwidth allocation. Instead of waiting for a response for the policy server, the Oracle USM immediately processes the call. When and if the policy server responds, the Oracle USM handles the response, but in all likelihood the priority calls has already been processed.

Enhanced Call Gapping

NMC provides flexibility by allowing a desired call-per-second (CPS) threshold to be achieved or surpassed by a predictable amount. Referred to as call gapping, this allows the Oracle USM to average the call rate and widen the period of a surge that would invoke NMC rules.

Without call gapping enabled, the NMC carries out a call gapping policy that monitors the arrival times between INVITEs, and then compares the arrival times to with the threshold. To

enable this, you set the **type** parameter to gap-rate, and then configure the **value** parameter with the maximum sustained rate of calls. The threshold is equal to 1/gap-rate value. However, this implementation means that if two calls arrive simultaneously at the Oracle USM, one of them might be rejected or diverted if it exceeds the threshold and the control rule is applied. This is the case even when the sustained call rate does not exceed the control rule.

To resolve this, call gapping uses two parameters that form part of an calculation the Oracle USM performs for applying NMC rules. Using the current time, the time of the last call gapped, the call counter value (tracked internally by the Oracle USM), the CPS value for the gap-rate control rule, and the values of the new parameters, the Oracle USM performs calculations that determine whether or not to apply the control rule.

About the Call Gapping Algorithm

The Oracle USM employs this leaky bucket algorithm to enforce calls per second. It smooths the call rate over a defined window of time to protect against surges. The values used for the calculation are:

- A—Calls per second; configure by setting the **type** parameter to gap-rate, and the **value** parameter to the CPS you want enforced
- m—Maximum counter value; must be greater than 0
- W—Window size; must be greater than
- deltaT—Time between allowed calls matching an NM control rule

The calculation is performed as follows, with the noted results:

- 1 + m - m*A*deltaT/W =< M—Means the call is allowed
- 1 + m - m*A*deltaT/W >M—Means that NMC rules are applied

Note the following:

- Setting the counter value and the window size to the same values guarantees that the processed CPS load will not exceed the desired CPS target.
- As the counter value becomes greater than the window size value, rejection rate will drop and the desired CPS threshold is not guaranteed.
- Increasing the window size results in a lower rejection rate when the attempted CPS is the same as the desired CPS; as the attempted CPS rate increases, rejection rates increase at a steeper rate.
- If either the count rate or the window size is set to 0, then the Oracle USM reverts to call gapping behavior it uses when the relevant parameters are not configured.

Network Management Control Configuration

In order use the network management controls feature, you need to set control rules and then enable their application on a per-realm basis. This section shows you how to set up those configuration.

Configuring an Individual Control Rule

To configure individual network management control rule:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

   ```
   ORACLE(session-router)# net-management-control
   ```

4. **name**—Enter the name of this network management control rule; this value uniquely identifies the control rule. There is no default for this parameter.

5. **state**—Enable or disable this network management control rule. The default value is **enabled**. The valid values are:

   - enabled | disabled

6. **destination-identifier**—Enter the call classification key. This parameter specifies information about the destination, which can be an IP address, an FQDN, a destination (called) number, or destination prefix. You can wildcard characters in the classification key using the carat symbol (^).

   You can enter special characters in the **destination-identifier** parameter. You can enter characters such as the plus-sign (+), the asterisk (*), the pound sign (#), capital letter A (A), capital letter B (B), capital letter C (C), capital letter D (D), lowercase letter p (p), lowercase letter w (w).

   This parameter can accommodate a list of entries so that, if necessary, you can specify multiple classification keys. You can edit the list of classification keys using the ACLI **add-destination-identifier** and **remove-destination-identifier** commands from within the network management controls configuration.

7. **type**—Enter the control type that you want to use. The valid values are:

   - **GAP-RATE**—Controls the maximum sustained rate of calls that match the classification key.
   - **GAP-PERCENT**—Controls the percentage of calls matching the classification key you set for the control rule.
   - **PRIORITY**—Exempts calls to a destination (like 911) from local network management controls. Use this value if you want to enable Priority Call Exemption from Policy Server Approval.

8. **value**—When you set the control type to either GAP-RATE or GAP-PERCENT, enter the maximum sustained rate of calls or the percentage of calls to which you want the control rule applied. The default value is zero (**0**). The valid values are:

   - **GAP-RATE**—Maximum is 2147483647 (which you can set by entering -1 as the value, an option provided for ease of use)
     - Using the minimum value (0) means that the Oracle USM treats all calls
     - Using the maximum value means that the Oracle USM treats no calls
   - **GAP-PERCENT**—Maximum is 100
     - Using the minimum value (0) means that the Oracle USM treats no calls
     - Using the maximum value (100%) means that the Oracle USM treats all calls

9. **treatment**—Enter the treatment method that you want to use. The default value is **none**. The valid values are:

   - **reject**—The Oracle USM rejects the call.

- **divert**—The Oracle USM routes the call to the location you specify in the control rule's next hop parameter.

10. **next-hop**—Enter the next hop for the Oracle USM to use when the treatment method is **DIVERT.** The valid values are:
    - hostname(:port)
    - IP address(:port)
    - Name of a valid, configured session agent
    - Name of a valid, configured session agent group—When you set this parameter to a session agent group, you must specify that it is a session agent group by prepending the name of the group with either SAG: or sag:. For example, the entry for a session agent group with Group2 as its name would be SAG:Group2 or sag:Group2.
11. **realm-next-hop**—Enter the realm identifier to designate the realm of the next hop when the treatment type is **DIVERT**.
12. **protocol-next-hop**—Enter the signaling protocol for the next hop when the treatment type is **DIVERT**.
13. **status-code**—Enter the SIP response code that you want the Oracle USM to use when the treatment method is **REJECT**. The default value is **503** (Service Unavailable). The valid range is:
    - Minimum—1
    - Maximum—699
14. **cause-code**—Enter the Q.850 cause code that you want the Oracle USM to use when the treatment method is **REJECT**. The default value is **63** (Service or option not available). The valid range is:
    - Minimum—1
    - Maximum—999999999

      For a SIP call, the Oracle USM replaces the cause code in the Reason header of the SIP response.

      For a H.323 call, the Oracle USM converts the cause code to a Q.931 cause code in the Q.931 Cause IE in the releaseComplete message.

Enabling Enhanced Call Gapping

Enhanced NMC call gapping uses new configuration parameters to the network management controls configuration:

- **gap-rate-max-count**—Maximum count that triggers the application of network management control rule if it is exceeded. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
- **gap-rate-window-size**—Length of time in seconds used for the gapping rate calculation. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999

For this feature to behave as intended, you also need to set the **type** parameter to **gap-rate**, and set the **value** parameter to the maximum sustained rate of calls that you want to support.

To configure NMC call gapping enhancements:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **net-management-control** and press Enter.

```
ORACLE(session-router)# net-management-control
```

To add support to a pre-existing network management control configuration, use the ACLI **select** command to choose the configuration you want to edit.

- **gap-rate-max-count**—Maximum count that triggers the application of network management control rule if it is exceeded. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
    Along with the current time, the last time of a gapped call, the call counter value, the CPS value, and the gap-rate-window-size value, the Oracle USM uses **gap-rate-max-count** as a measurement to determine if a control rule will be applied.
- **gap-rate-window-size**—Length of time in seconds used for the gapping rate calculation. The default value is zero (**0**). The valid range is:
  - Minimum—0
  - Maximum—999999999
    Along with the current time, the last time of a gapped call, the call counter value, and the CPS value, the Oracle USM uses the **gap-rate-window-size** value to calculate whether the maximum count is within allowable limits.

1. Save and activate your configuration.

Applying a Network Management Control Rule to a Realm

Once you have configured network management control rules, you can enable their use on a per-realm basis.

To apply a network management control rule to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   If you are enabling network management controls for a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **net-management-control**—Set this parameter to **enabled** to apply network control rules in this realm. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

3147 - Emergency Call Fallback

You can enable the rejection of priority calls per a configured SIP interface. When this option is enabled, the Oracle USM does not allow priority calls through that SIP interface, and the call is rejected with a 380 response.

If the send-380-response parameter in the SIP interface is configured, this string is provided as a reason in the 380 message. If this parameter is not populated, the default message priority calls not allowed is sent.

Emergency Call Fallback Configuration

To enable emergency call fallback, you must select an option for the SIP-interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ACMEPACKET(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(session-router)# sip-interface
   ACMEPACKET(sip-interface)#
   ```

4. **options**—Type **disallow-priority-calls** to send a 380 response to priority calls.

   ```
   ACMEPACKET(session-router)# options disallow-priority-calls
   ```

5. Save and activate your configuration.

Accounting Configuration for QoS

This section explains how to configure the account configuration and account servers so you can use the Oracle USM in conjunction with external RADIUS (accounting) servers to generate CDRs and provide billing services requires.

QoS Accounting Configuration

To configure the account configuration and account servers:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **account-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(session-router)# account-config
ORACLE(account-config)#
```

4. To configure account server parameters (a subset of the account configuration parameters, type **account-servers** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(account-config)# account-servers
   ORACLE(account-server)#
   ```

   The following example shows both the account config and account server parameters.

   ```
   account-config
           hostname                        acctserver1
           port                            1813
           strategy                        Hunt
           state                           enabled
           max-msg-delay                   60
           max-wait-failover               100
           trans-at-close                  disabled
           generate-start                  OK
           generate-interim
                                           OK
                                           Reinvite-Response
   account-server
                   hostname                        192.168.2.2
                   port                            1813
                   state                           enabled
                   min-round-trip                  100
                   max-inactivity                  100
                   restart-delay                   100
                   bundle-vsa                      enabled
                   secret                          testing
                   NAS-ID                          acme-accounting
   last-modified-date              2005-01-15 02:23:42
   ```

Account Configuration

You set the account configuration parameters to indicate where you want accounting messages sent, when accounting messages you want them sent, and the strategy you want used to select account servers.

To configure the account configuration:

1. **hostname**—Enter a name for the host associated with the Oracle USM in hostname (FQDN) format. The default value is the name of the local host.

   The value you enter here must match the configured physical interface's operation type **control** or **maintenance**, to determine on which network to send RADIUS messages.

2. **port**—Enter the number of the UDP port associated with the Oracle USM from which RADIUS messages are sent.The default value is **1813**. The valid range is:

   - Minimum—1025
   - Maximum—65535

3. **strategy**—Indicate the strategy you want used to select the accounting servers to which the Oracle USM will send its accounting messages. The default value is **hunt**. The following table lists the available strategies:

   - **hunt**—Selects accounting servers in the order in which they are listed.

If the first accounting server is online, working, and has not exceeded any of the defined constraints, all traffic is sent to it. Otherwise the second accounting server is selected. If the first and second accounting servers are offline or exceed any defined constraints, the third accounting server is selected. And so on through the entire list of configured servers

- **failover**—Uses the first server in the list of predefined accounting servers until a failure is received from that server. Once a failure is received, it moves to the second accounting server in the list until a failure is received. And so on through the entire list of configured servers.
- **roundrobin**—Selects each accounting server in order, distributing the selection of each accounting server evenly over time.
- **fastestrtt**—Selects the accounting server that has the fastest round trip time (RTT) observed during transactions with the servers (sending a record and receiving an ACK).
- **fewestpending**—Selects the accounting server that has the fewest number of unacknowledged accounting messages (that are in transit to the Oracle USM).

4. **state**—Enable this parameter if you want the account configuration active on the system. Disable it if you do not want the account configuration active on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
5. **max-msg-delay**—Indicate the length of time in seconds that you want the Oracle USM to continue trying to send each accounting message. During this delay, the Oracle USM can hold a generic queue of 4096 messages. The default value is **60**.
   - Minimum—zero (0)
   - Maximum—4294967295
6. **max-wait-failover**—Indicate the maximum number of accounting messages the Oracle USM can store its message waiting queue for a specific accounting server, before it is considered a failover situation.

   Once this value is exceeded, the Oracle USM attempts to send it accounting messages, including its pending messages, to the next accounting server in its configured list. The default value is **100**. The valid range is:
   - Minimum—1
   - Maximum—4096
7. **trans-at-close**—Disable this parameter if you do not want to defer the transmission of message information to the close of a session. Enable it if you want to defer message transmission. The default value is **disabled**. The valid values are:
   - **disabled**—The Oracle USM transmits accounting information at the start of a session (Start), during the session (Interim), and at the close of a session (Stop). The transmitted accounting information for a single session might span a period of hours and be spread out among different storage files.
   - **enabled**—Limits the number of files on the Oracle USM used to store the accounting message information for one session. It is easiest to store the accounting information from a single session in a single storage file.
8. **generate-start**—Select the type of SIP event that triggers the Oracle USM to transmit a RADIUS Start message. The default value **is** ok. The valid values are:
   - **start**—RADIUS Start message should not be generated

- **invite**—RADIUS Start message should be generated once the Oracle USM receives a SIP session INVITE.
- **ok**—RADIUS Start message is generated once the Oracle USM receives an OK message in response to an INVITE.

9. **generate-interim**—Retain the default value **reinvite-response** to cause the Oracle USM to transmit a RADIUS Interim message. (A RADIUS Interim message indicates to the accounting server that the SIP session parameters have changed.)

   To disable interim message generation, enter a pair of quotes as the value for this parameter. Otherwise, select one or more than one of the following values:

   - **ok**—RADIUS Interim message is generated when the Oracle USM receives an OK message in response to an INVITE.
   - **reinvite**—RADIUS Interim message is generated when the Oracle USM receives a SIP session reINVITE message.
   - **reinvite-response**—RADIUS Interim message is generated when the Oracle USM receives a SIP session reINVITE and responds to it (for example, session connection or failure).
   - **reinvite-cancel**—RADIUS Interim message is generated when the Oracle USM receives a SIP session reINVITE, and the Reinvite is cancelled before the Oracle USM responds to it.

10. **account-server**—Create the account server list to store accounting server information for the account configuration. Each account server can hold 100 accounting messages.

    Account server entries are specific to the account configuration. They cannot be viewed or accessed for editing outside of the account configuration.

> **Note:**
>
> RADIUS will not work if you do not enter one or more servers in a list.

Account Server

You must establish the list of servers to which the Oracle USM can send accounting messages.

1. **hostname**—Name of the host associated with the account server as an IP address.
2. **port**—Enter the number of the UDP port associated with the account server to which RADIUS messages are sent. The default value is **1813**. The valid range is:
   - Minimum—1025
   - Maximum—65535
3. **state**—Enable or disable the account servers on the system. The default value is **enabled**. The valid values are:
   - enabled | disabled
4. **min-round-trip**—Indicate the minimum round trip time of an accounting message in milliseconds. The default value is **250**. The valid range is:
   - Minimum—10
   - Maximum—5000

A round trip consists of the following:

The system sends an accounting message to the account server.

The account server processes this message and responds back to the Oracle USM.

If the **fastest RTT** is the **strategy** for the account configuration, the value you enter here can be used to determine an order of preference (if all the configured account servers are responding in less than their minimum RTT).

5. **max-inactivity**—Indicate the length of time in seconds that you want the Oracle USM with pending accounting messages to wait when it has not received a valid response from the target account server. The default value is **60**. The valid range is:

   - Minimum—1
   - Maximum—300

   Once this timer value is exceeded, the Oracle USM marks the unresponsive **account server** as disabled in its failover scheme. When a server connection is marked as inactive, the Oracle USM attempts to restart the connection and transfers pending messages to another queue for transmission. RADIUS messages might be moved between different **account server**s as servers become inactive or disabled.

6. **restart-delay**—Indicate the length of time in seconds you want the Oracle USM to wait before resending messages to a disabled account server. The default value is **30**. The valid range is:

   - Minimum—1
   - Maximum—300

7. **bundle-vsa**—Retain the default **enabled** if you want the account server to bundle the VSAs within RADIUS accounting messages. Enter **disabled** if you do not want the VSAs to be bundled. (Bundling means including multiple VSAs within the vendor value portion of the message.) The valid values are:

   - enabled | disabled

   In a bundled accounting message, the RADIUS message type is vendor-specific, the length is determined for each individual message, and the vendor portion begins with a 4-byte identifier, and includes multiple vendor type, vendor length, and vendor value attributes.

8. **secret**—Enter the secret passed from the account server to the client in text format. Transactions between the client and the RADIUS server are authenticated by the shared secret; which is determined by the source IPv4 address of the received packet.

9. **NAS-ID**—Enter the NAS ID in text format (FQDN allowed). The account server uses this value to identify the Oracle USM for the transmittal of accounting messages.

   The remote server to which the **account configuration** sends messages uses at least one of two potential pieces of information for purposes of identification. The Oracle USM accounting messages always includes in the first of these:

   - Network Access Server (NAS) IP address (the IP address of the Oracle USM's SIP proxy)
   - **NAS ID (t**he second piece of information) provided by this value. If you enter a value here, the NAS ID is sent to the remote server.

Whitelists for SIP

Oracle USM by default ignores and passes-through unknown SIP headers and URI parameters. However, some operators require that the Oracle USM only accept messages with headers and URI parameters complying with those supported by their internal equipment. This section describes the use of whitelists to control unknown headers and parameters in request and response traffic.

What is a Whitelist

A whitelist is an approved list of entities for which equipment provides particular privileges, access, and recognition. The Oracle USM can use configured whitelist profiles to control and accept specific inbound SIP headers and URI parameters that are being passed-through the Oracle USM. When you configure this feature, the Oracle USM rejects requests not matching the configured profile, or removes the unspecified headers or URI parameters not in the configured profile.

Whitelists Configuration

You can configure whitelist profiles or rules that allow the Oracle USM to only accept inbound SIP headers and URI parameters that are configured in this whitelist, using the parameter **allowed-elements-profile**. You can configure the settings for this parameter using the ACLI interface at session-router>enforcement-profile. Since the **enforcement-profile** object also pertains to session agents, realms, and SIP interfaces, you can also apply the profiles you configure to these remote entities using the ACLI interface at session-router>session-agent, session-router>sip-interface, and media-manager>realm-config.

In the following configuration example, it is assumed that your baseline configuration passes SIP traffic, with the Oracle USM in the role of an Access SBC. Use this procedure to configure a whitelist for the session router and optionally apply the specific whitelists to the session agent and SIP interface, as well as the media manager's realm configuration.

To configure a whitelist for the session router:

1. Access the **allowed-elements-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# allowed-elements-profile
   ORACLE(allowed-elements-profile)#
   ```

2. **name**— Enter a unique name for the whitelist you are creating. This name can be referenced when configuring the enforcement-profiles for session-agent, SIP interface, and realm-config.

   ```
   ORACLE(allowed-elements-profile)# name whitelist1
   ```

3. **description**— Enter a description that explains the purpose of creating this whitelist. You can use any alpha-numeric characters when entering the description.

   ```
   ORACLE(allowed-elements-profile)# description Basic Whitelist
   ```

4. Navigate to the **rule-sets** configuration element to specify the rules to match against specific incoming SIP headers and/or URI parameters.

   ```
   ORACLE(allowed-elements-profile)# rule-sets
   ORACLE(rule-sets)#
   ```

5. **unmatched-action**— Select the action for the Oracle USM to perform when a header does not exist in an incoming message. The default value is **reject**. The valid values are:

   - **reject** — Rejects all incoming messages that do not contain a header.
   - **delete** — Deletes all incoming message that do not contain a header.

   > **Note:**
   >
   > This parameter applies to non-matching header names only (not non-matching URI parameters).

   ```
   ORACLE(rule-sets)# unmatched action delete
   ```

6. **msg-type** — Specify the type of messages for which the Oracle USM applies this whitelist configuration. The default value is **any**. The valid values are:

   - **any** — Applies to all incoming messages.
   - **request** — Applies to only incoming REQUEST messages.
   - **response** — Applies to only incoming RESPONSE messages.

   ```
   ORACLE(rule-sets)# msg-type any
   ```

7. **methods** — Enter the packet method(s), separated by a comma, for which this whitelist is enforced. Packet methods include, INVITE, OPTIONS, ACK, BYE, etc. If this field is left blank, the whitelist applies to all packet methods. You can enter up to a maximum of 255 characters.

   ```
   ORACLE(rule-sets)# methods INVITE,ACK,BYE
   ```

8. **logging** — Select whether or not an incoming message is written to a log file called **matched.log** when the message contains an element not specified in the whitelist. The default value is **disabled**. The valid values are:

   - enabled | disabled

   ```
   ORACLE(rule-sets)# logging enabled
   ```

   The **matched.log** contains information about the timestamp, received/sent Oracle USM network-interface, IP address/port from which it was received or being sent from, and which peer IP address/port it was received from or sent to. The log also specifies the request URI (if applicable), and the From, To, and Contact headers in the message, as well as which rule triggered the log action. An example of the log output of the **matched.log** file is as follows:

   ```
   Dec 17 14:17:54.526 On [0:0]192.168.1.84:5060 sent to 192.168.1.60:5060
   allowed-elements-profile[whitelist1(reject)]
   INVITE sip:service@192.168.1.84:5060 SIP/2.0
   From: sipp <sip:+2125551212@192.168.1.60:5060>;tag=3035SIPpTag001
   To: sut <sip:service@192.168.1.84>
   Contact: sip:sipp@192.168.1.60:5060
   ```

   The **header-rule** object consists of 6 parameters that make up the header-rule:

   - header-name
   - unmatched-action
   - allow-header-param
   - allow-uri-param

- allow-uri-user-param
- allow-uri-header-name

  You can configure an unlimited number of header-rules on the Oracle USM that you can apply to your network. Use the following parameters to configure a header-rule that the Oracle USM uses to control which incoming messages it allows.

9. **header-rule** — This object allows you to configure, as part of the whitelist, multiple parameters which make up the header rule that the Oracle USM allows from incoming messages. Header-rules do NOT have to be in any specific order. The system prompt changes to let you know that you can begin configuring individual parameters for this object.

   ```
   ORACLE(rule-set)# header-rule
   ORACLE(allowed-header-rule)#
   ```

10. **header-name** — Enter the name of the header in the whitelist that the Oracle USM allows from incoming messages. It is case-insensitive and supports abbreviated forms of header names. For example, “Via”, “via”, or “v” all match against the same header. A header name of “request-uri” refers to the request URI of requests, while a header name of * applies to any header-type not matched by any other header-rule. The default value is *. This default value provides the ability to have header-rules for commonly known headers that remove unknown parameters, but leave unknown headers alone.

    ```
    ORACLE(allowed-header-rule)# header-name Contact
    ```

11. **unmatched-action** — Select the action for the Oracle USM to perform when an incoming header’s parameters do not match the relevant allowed parameters specified for this header-name. The default value is **reject**. The valid values are:

    - **reject** — Rejects all incoming messages that have header parameters that do not match the parameters specified in this header-name.
    - **delete** — Deletes all incoming messages that have header parameters that do not match the parameters specified in this header-name.

    > **Note:**
    >
    > This parameter applies to non-matching header names only (not non-matching URI parameters).

    ```
    ORACLE(allowed-header-rule)# unmatched-action delete
    ```

12. **allow-header-param** — Enter the header parameter that the Oracle USM allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). The default value is *, which allows all header parameters to pass through. If you leave this field empty, no header parameters are allowed.

    ```
    ORACLE(allowed-header-rule)# allow-header-param *
    ```

13. **allow-uri-param** — Enter the URI parameter that the Oracle USM allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). The default value is *, which allows all URI parameters to pass through. If you leave this field empty, no URI parameters are allowed.

    ```
    ORACLE(allowed-header-rule)# allow-uri-param *
    ```

14. **allow-uri-user-param** — Enter the URI user parameter that the Oracle USM allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). The default value is *, which allows all URI user parameters to pass through. If you leave this field empty, no URI user parameters are allowed.

    ```
    ORACLE(allowed-header-rule)# allow-uri-user-param *
    ```

15. **allow-uri-header-name** — Enter the URI header name that the Oracle USM allows from the headers in incoming messages. You can enter up to 255 characters, including a comma (,), semi-colon (;), equal sign (=), question mark (?), at-symbol (@), backslash (\), or plus sign (+). The default value is *, which allows all URI header name parameters to pass through. If you leave this field empty, no URI header name parameters are allowed.

    ```
    ORACLE(allowed-header-rule)# allow-uri-header-name *
    ```

16. Save your work using the **done** command.

Configuration Exception

There are specific instances in incoming Request-URI messages where the Oracle USM ignores specific parameters and automatically adds header-rules.

In a Request-URI, all parameters are URI parameters, and URI headers are not allowed. If you define values for the “allow-header-param”, “allow-uri-header-name”, and “allow-uri-param”, the Oracle USM ignores these parameters in the Request-URI. Instead the Oracle USM automatically adds header-rules for incoming “Via”, “From”, To”, “Call-ID”, and “CSeq” messages. These are explicit header rules and cannot be deleted. Each header-rule in a Request-URI has parameters populated with the value of *. If required, a user can change the header-rule parameter values with the values identified in the following table.

Header Rule	Applicable Parameter	Required Value(s)
Via	allow-header-param	• branch • received • rport
From	allow-header-param	• tag
To	allow-header-param	• tag
Call-ID	allow-header-param	No restrictions
CSeq	allow-header-param	No restrictions

Verify Whitelist Configuration

After you have configured and saved a whitelist on the Oracle USM, you can use the existing verify-config command at the top level prompt to verify the saved configuration: For example:

```
ORACLE# verify-config
```

This command checks for errors in the Oracle USM configuration. Whitelist configuration errors specifically related to the enforcement-profile object also display in the output of this command if applicable. The whitelist configuration errors display if any references to the allowed-element-profiles are improperly configured. If errors exist, the following message displays:

```
--------------------------------------------------------------------
ERROR: enforcement-profile [ep] contains a reference to an allowed-enforcement-
```

```
profile [abc] that does not exist
--------------------------------------------------------------------
```

How Whitelists Work

Whitelists allow you to customize which SIP signaling messages to allow into your network and which messages to reject or delete. In the flow of SIP traffic to/from the Oracle USM, the Oracle USM matches any received request or response, in or out of a dialog against the configured allowed list, and rejects or deletes the non-matching element based on the actions specified in the whitelist configuration.

For responses, the Oracle USM does not reject the message if a header or parameter is not found in the allowed list, even if the action is set to reject. Instead it deletes the offending parameter or header. In addition, if the message is a request of the method type ACK, PRACK, CANCEL or BYE, it deletes all unmatched elements, but does not reject the request, even if the action was configured to reject.

The whitelist verification performs for any method; however you can narrow this list to operate only on specific methods by defining them in the **methods** parameter of the configuration.

Whitelist verification occurs when a request or response is received by the Oracle USM, but only after the Oracle USM has processed the inbound header manipulation rule (HMR), network management controls (NMC), Resource-Priority header (RPH), and monthly-minutes checking.

The Oracle USM responds to requests which have non-matching headers or parameters configured with an action of reject, with a "403 Forbidden" response by default. You can use a local-response event, **allowed-elements-profile-rejection**, to override the default reject status code and reason phrase.

The configured whitelist operates transparently on headers that have multiple URIs or multiple header values within a single header (header values separated by a comma).

Parameter parsing operates only on parameters that it can identify. For parameters that can not be parsed, for example an invalid URI (e.g. <sip:user@host.com&hp=val>), the Oracle USM ignores this URI header parameter value of "hp" since it is not contained within a valid URI. Even though it would appear to be a URI header parameter, URI headers must come after URI parameters. Parameter matching does not occur if the headers and parameters in the URI are not well-formed. The Oracle USM does not remove the parameter since it cannot identify it.

Whitelist Learning

You can build your whitelist configuration based on the learning capabilities of the Oracle USM. When you enable the Oracle USM learning mode, it acquires the knowledge of the allowable elements (headers and parameters) currently incoming to your network. The Oracle USM collects the information about the headers received and the parameters that exist within each header. The information continues to be gathered until you disable the learning mode.

Once you disable the learning mode, the Oracle USM prompts you to enter a name for the allowed-elements-profile. If the profile name you entered does not exist, the captured information is written to the new allowed-elements-profile configuration. The administrator can then make changes to the configuration as applicable, save the configuration, and apply it to a logical remote entity.

The new allowed-elements-profile does not contain any wildcard rules. The Oracle USM cannot generate wildcard headers and parameters during the learning mode. The Methods object is populated from the list of methods seen by the Oracle USM while learning.

> **Note:**
>
> Oracle recommends running the learning mode during off-peak and/or light traffic times. This mode can operate in conjunction with the execution of an allowed-elements-profile. The learning occurs just before any configured allowed-elements-profile configuration.

Whitelist Learning Configuration

The ACLI interface provides two commands that allow a Superuser to start and stop whitelist learning on the Oracle USM:

Command	Description
start <argument> <options>	Starts whitelist learning on the Oracle USM. You must specify the argument learn-allowed-elements with this command to start the learning operation. Optionally, you can use method, msg-type, and params after the argument.
stop <argument> <identifier>	Stops the whitelist learning on the Oracle USM and writes the learned configuration to the editing configuration on the Oracle USM where it is saved and activated. You must specify the argument learn-allowed-elements with this command to stop the learning operation. You must specify a unique identifier that identifies the allowed-elements-profile name. If you specify an identifier name that already exists as a profile, the ACLI returns an error message and prompts you to enter a different name.

You can use these commands at the top level ACLI prompt as required on the Oracle USM.

You use these commands with the argument, learn-allowed-elements to start/stop the whitelist learning feature. By default, the learning mode creates a single rule-set under which all of the headers and their respective parameters are stored.

For example:

```
ORACLE# start learn-allowed-elements
Learning mode for allowed-elements-profile started.
```

In the above example, **start** is the top level ACLI command and **learn-allowed-elements** is the operation being performed.

Optionally, you can specify [method], [msg-type], and [params] in any order, for the Oracle USM to learn specific rule-set elements from incoming messages and save them to the whitelist configuration.

For example:

```
ORACLE# start learn-allowed-elements method msg-type params
```

The **method** option creates a new rule-set per unique method. The **msg-type** option creates a new rule-set per unique message-type seen. The **params** option performs URI and header parsing to examine parameters within the message. By default, parameter parsing is disabled.

To start the whitelist learning feature:

In Superuser mode, at the top level ACLI prompt, type `start learn-allowed-elements` and press Enter.

```
ORACLE# start learn-allowed-elements
```

The following message displays:

```
Learning mode for allowed-elements-profile started.
```

To specify the elements of rule-sets for whitelists:

In Superuser mode, at the top level ACLI prompt, type `start learn-allowed-elements method msg-type params` and press Enter.

```
ORACLE# start learn-allowed-elements method msg-type params
```

The following message displays:

```
Learning mode for allowed-elements-profile started.
```

> **Note:**
>
> If you try to start a whitelist learning operation while another learning operation is already running, the following message displays:
>
> ```
> Learning mode restarted without saving
> Learning mode for allowed-elements-profile started.
> ```

To stop the whitelist learning feature:

In Superuser mode, at the top level ACLI prompt, type **stop learn-allowed-elements <identifier>**, where <identifier> is the allowed-elements-profile name, and press Enter.

```
ORACLE# stop learn-allowed-elements whitelist1
```

The following message displays:

```
Learning mode for allowed-elements-profile stopped.
```

If you specify an identifier name that already exists as a profile, the ACLI returns an error message and prompts you to enter a different name.

Rejected Messages Monitoring

Whitelists, when configured on the Oracle USM, control whether or not the Oracle USM allows unknown headers and URI parameters to be accepted in incoming request and response traffic. When the Oracle USM rejects messages according to the whitelist, the rejected messages are logged to a file called “matched.log” if logging is set to enabled. You can open and view the log when required to view the rejected messages.

In addition to the rejected messages being logged to the “matched.log” file, the rejected messages are also sent through a burst counter that keeps track of the amount of messages rejected. You can enter the show sipd to display the number of rejected messages. The counter is titled Rejected Message.

14
Static Flows

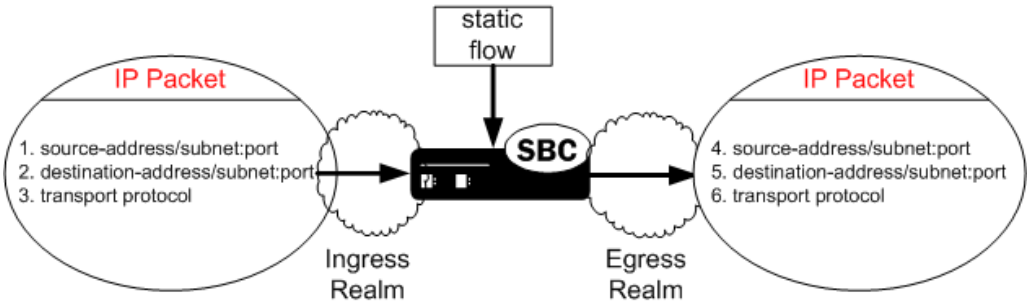
This chapter describes the Oracle USM's static flows feature. Static flows allow network traffic that matches specific criteria to pass through the Oracle USM unrestricted. Static flows are unidirectional. This feature lets you steer traffic toward a particular destination based on its original characteristics. Static flows can range from being widely accessible to very restrictive, depending on the values you establish. Static flows are used for transporting a variety of signaling messages through the Oracle USM to achieve vendor interoperability. The Oracle USM supports the following types of Static flows:

- IPv6 to IPv6 flows
- IPv4 to IPv6 flows
- IPv4 to IPv4 flows

> **Note:**
>
> Traffic that traverses the Oracle USM in two directions, such as ICMP requests and responses, requires static flows configured for both directions.

About Static Flows

The static flow element explicitly writes entries into the IP routing table. These entries are persistent and are not deleted as calls are set up and broken down. Refer to the following diagram to understand how a static flow works.



A static flow entry watches for traffic with specific criteria on a specified ingress realm; that traffic consists of the following criteria:

- The packet enters the Oracle USM on the specified ingress realm.
- The packet contains matching source address, subnet, and port criteria, field 1.
- The packet contains matching destination address, subnet, and port criteria, field 2.
- The packet contains a matching transport protocol, field 3.

If the above conditions are met, then the Oracle USM does the following:

- The IPv4 traffic is forwarded out of the Oracle USM on the specified egress realm.
- The configured source address, subnet, and port criteria are written to the exiting packet, field 4.
- The configured destination address, subnet, and port criteria are written to the exiting packet, field 5.
- The original transport protocol and its contents remain unchanged as the packet exits into the egress realm.

IPv6 / IPv4 Translations

The ingress or egress traffic type, whether IPv4 or IPv6, must match the configuration of the realm where attached, as **in-realm-id** or **out-realm-id**. A realm and IP version configuration mismatch results in an error message and log entry at error level.

IPv6 to IPv4 flows exit the Oracle USM with the prefix `::fff:0:0:0/96`. They may be written as `::ffff:0:a.b.c.d`, where a.b.c.d refers to an IPv6-enabled node.

While IPv4 addresses can be translated into IPv6 addresses, IPv6 address can not be translated to IPv4.

About Network Address Translation ALG

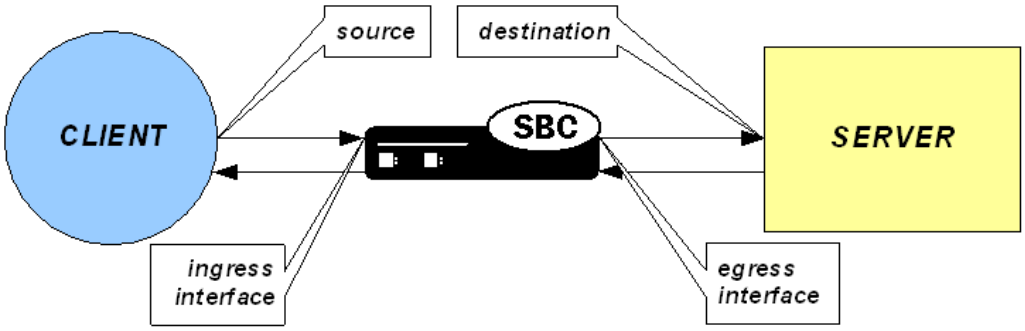
The Oracle USM supports Network Address and Port Translation (NAPT) and Trivial File Transfer Protocol (TFTP) functionality over media interfaces, collectively known as Network Address Translation (NAT) ALG. The NAT ALG feature is implemented as an extension of the static flow feature.

In some applications, the Oracle USM acts as an intermediary device, positioned between endpoints located in an access network and application servers located in a backbone network. The Oracle USM 's NAT ALG feature enables these endpoints to use non-VoIP protocols, such as TFTP and HTTP, to access servers in a provider's backbone network to obtain configuration information.

NAT ALG parameters support RTC and can be dynamically reconfigured. The active NAT ALG configuration can be replicated on the standby SD in an HA configuration.

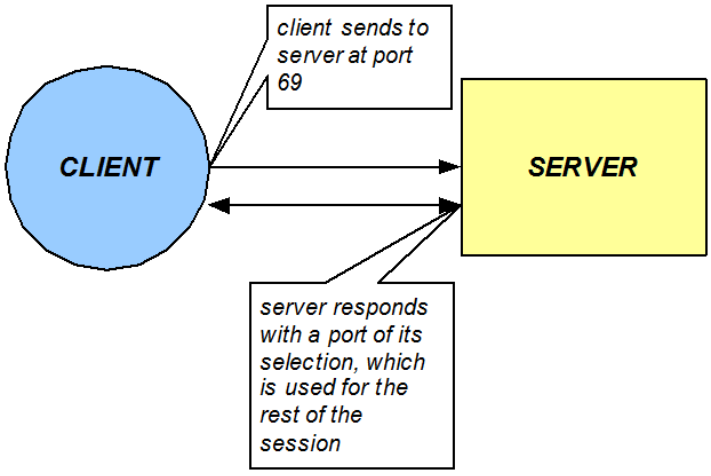
NAPT

The NAPT ALG functionality is the same as that found in commercially available enterprise and residential NAT devices. The Oracle USM watches for packets entering a media interface that match source and destination IP address criteria. Matching packets are then redirected out of the egress interface, through a specified port range, toward a destination address.

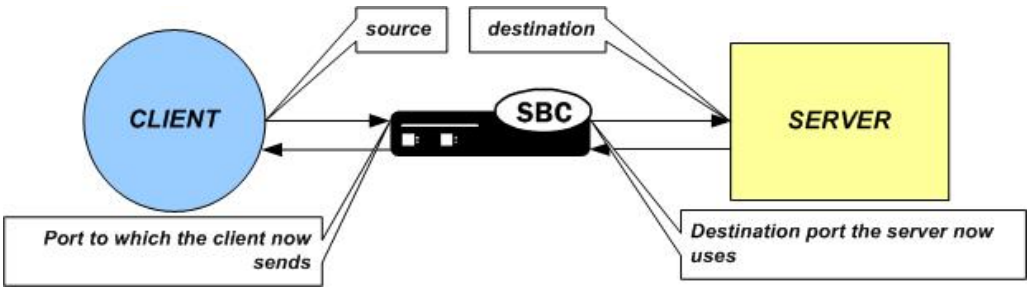


TFTP

The TFTP ALG is implemented as an extension of the NAT ALG. It works slightly differently than traditional NAPT. In a TFTP session, the first packet is sent from a source endpoint to port 69 on the TFTP server. The TFTP server responds from another port. This port, from which the TFTP response originates, is used for the remainder of the TFTP session.



To act as a TFTP ALG, the Oracle USM will latch on the first return packet from the server to learn the server's port. The ingress-side destination port of the Oracle USM is changed to reflect the new communications port for the TFTP session. This process takes place without any user intervention.



Configuring Static Flows

This section explains how to configure static flows. It also provides sample configurations for your reference. You can configure static flows with or without NAT ALG. If you configure static flows with NAT ALG, you can choose NAPT or TFTP as the ALG type.

Basic Static Flow Configuration Overview

This section outlines the basic static flow configuration, without NAT ALG. You configure static flows by specifying ingress traffic criteria followed by egress re-sourcing criteria.

When configuring static flows, the following conventions are used:

- An address of 0.0.0.0 matches all addresses. This token is used as the wildcard for both IPv4 and IPv6 static flows
- Enclose the address portion of an IPv6 address in brackets: `[7777::11]/64:5000`
- Not specifying a port implies all ports.
- Not specifying a subnet mask implies a /32, matching for all 32 bits of the IPv4 address , or a /128 matching for all 128 bits of the IPv6 address.

1. Set the static flows' incoming traffic-matching criteria. First set the ingress realm where you expect to receive traffic that will be routed via a static flow. Second, set the traffic's source IP address, source subnet, and source port or port range criteria. Third, set the traffic's destination IP address, destination subnet, and destination port criteria. This is usually an external address on theOracle USM.
2. Set the criteria that describes how traffic should be translated on the egress side of the Oracle USM. First set the egress realm where you want to send the traffic to be routed by this static flow. Second, set the traffic's source IP address, source subnet, and source port or port range criteria. This is usually an external address on the Oracle USM. Third, set the traffic's destination IP address, destination subnet, and destination port criteria.
3. Set the protocol this static flow entry acts upon. This type of packet, as the payload of the IP packet, remains untouched as traffic leaves the Oracle USM . Specifying a layer 4 protocol here acts as another criteria to filter against for this static flow.

   The combination of entries in the ingress realm, ingress source address, ingress destination address, and protocol fields must be unique. For bidirectional traffic, you need to define a separate static flow in the opposite direction.

Static Flow Configuration

This section describes how to configure the **static-flow** element using the ACLI.

The ingress IP address criteria is set first. These parameters are applicable to traffic entering the ingress side of the Oracle USM .

- **in-realm-id**—The access realm, where endpoints are located.
- **in-source**—The source network in the access realm where the endpoints exist. This parameter is entered as an IP address and netmask in slash notation to indicate a range of possible IP addresses.

- **in-destination**—The IP address and port pair where the endpoints send their traffic. This is usually the IP address and port on a Oracle USM physical interface that faces the access realm.

The egress IP address criteria is entered next. These parameters determine how traffic is re-sourced as it leaves the Oracle USM and enters the backbone network.

- **out-realm-id**—The backbone realm, where servers are located.
- **out-source**—The IP address on the physical interface of the Oracle USM where traffic exits the Oracle USM into the backbone realm. Do not enter a port for this parameter.
- **out-destination**—The IP address and port pair destination of the traffic. This is usually a server in the backbone realm.
- **protocol**—The protocol associated with the static flow. The protocol you choose must match the protocol in the IPv4 header. Valid entries are TCP, UDP, ICMP, ALL.

The type of NAT ALG, if any.

- **alg-type**—The type of NAT ALG. Set this to NAPT, TFTP, or none.

The port range for port re-sourcing as traffic affected by the NAT ALG exits the egress side of the Oracle USM is set next. (Not applicable if **alg-type** is set to none.)

- **start-port**—The starting port the NAT ALG uses as it re-sources traffic on the egress side of the Oracle USM .
- **end-port**—The ending port the NAT ALG uses as it re-sources traffic on the egress side of the Oracle USM .

The flow timers are set next. (Not applicable if **alg-type** is set to none.)

- **flow-time-limit**—Total session time limit in seconds. The default is 0; no limit.

> **Note:**
>
> Note that the static flow-time-limit must have a value larger than initial-guard-timer and subsq-guard-timer for static flows.

- **initial-guard-timer**—Initial flow guard timer for an ALG dynamic flow in seconds. The default is 0; no limit.
- **susbsq-guard-timer**—Subsequent flow guard timer for an ALG dynamic flow in seconds. The default is 0; no limit.

Finally, you can set the optional bandwidth policing parameter for static flows (with or without NAT ALG applied).

- **average-rate-limit**—Sustained rate limit in bytes per second for the static flow and any dynamic ALG flows. The default is 0; no limit.
  To configure static flow:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the **media-manager** path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **static-flow** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# static-flow
```

From this point, you can configure media policing parameters.

4. **in-realm-id**—Enter the ingress realm or interface source of packets to match for static flow translation. This in-realm-id field value must correspond to a valid identifier field entry in a **realm-config**. This is a required field. Entries in this field must follow the Name Format.
5. **in-source**—Enter the incoming source IP address and port of packets to match for static flow translation. IP address of 0.0.0.0 matches any source address. Port 0 matches packets received on any port. The port value has no impact on system operation if either ICMP or ALL is the selected protocol. This parameter takes the format:

   in-source <ip-address>[:<port>]

   The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
6. **in-destination**—Enter the incoming destination IP address and port of packets to match for static-flow translation. An IP address of 0.0.0.0 matches any source address. Port 0 matches packets received on any port. The port value has no impact on system operation if either ICMP or ALL is the selected protocol. The in-source parameter takes the format:

   in-destination <ip-address>[:<port>]

   The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
7. **out-realm-id**—Enter the defined realm where traffic leaving this NAT ALG exits theOracle USM .
8. **out-source**—Enter the egress IPv4 address. This is the IPv4 address of the network interface where traffic subject to the NAT ALG you are defining leaves the Oracle USM . Do not enter a port number for this parameter. The default value is **0.0.0.0**.
9. **out-destination**—Enter the IPv4 address and port number of the server or other destination to which traffic is directed. The default value is **0.0.0.0**. The valid port range is:

   - Minimum—0
   - Maximum—65535
10. **protocol**—Enter the protocol this NAPT ALG acts upon. The default value is **UDP**. The valid values are:

    - TCP | UDP | ICMP | ALL
11. **alg-type**—Enter the type of NAT ALG to use. The default value is **none**. The valid values are:

    - **none**—No dynamic ALG functionality
    - **NAPT**—Configure as NAPT ALG
    - **TFTP**—Configure as TFTP ALG
12. **start-port**—Enter the beginning port number of the port range that the Oracle USM allocates on the egress side for flows that this NAPT ALG redirects. The default value is **0**. The valid range is:

- Minimum—0, 1025
- Maximum—65535

13. **end-port**—Enter the ending port number of the port range that the Oracle USM allocates on the egress side for flows that this NAPT ALG redirects. The default value is **0**. The valid range is:
    - Minimum—0, 1025
    - Maximum—65535
14. **flow-time-limit**—Enter the total time limit for a flow in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
15. **initial-guard-timer**—Enter the initial guard timer value in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
16. **subsq-guard-timer**—Enter the subsequent guard timer value in seconds. A value of **0** means there is no limit. The valid range is:
    - Minimum—0
    - Maximum—999999999
17. **average-rate-limit**—Enter a maximum sustained rate limit in bytes per second. The default value is **0**; no limit. The valid range is:
    - Minimum—0
    - Maximum—125000000

    The following example shows a **static-flow** configuration element configured for a NAPT ALG.

```
in-realm-id                 access
in-source                   172.16.0.0/16
in-destination              172.16.1.16:23
out-realm-id                backbone
out-source                  192.168.24.16
out-destination             192.168.24.95:23
protocol                    TCP
alg-type                    NAPT
start-port                  11000
end-port                    11999
flow-time-limit             0
initial-guard-timer         60
subsq-guard-timer           60
average-rate-limit          0
```

Example Configuration: Bidirectional Static Flows

The configuration lines below present the configuration of two example static flows to be used for ICMP to a specific host through the Oracle USM.

The following lines present the example configuration for the access to core side.

```
static-flow
in-realm-id access
description
in-source 0.0.0.0
in-destination 10.1.215.63
out-realm-id core
out-source 10.2.214.63
out-destination 10.2.214.51
protocol ICMP
alg-type none
start-port 0
end-port 0
flow-time-limit 0
initial-guard-timer 60
subsq-guard-timer 60
average-rate-limit 0
```

The following lines present the example configuration for the core to access side.

```
static-flow
in-realm-id core
description
in-source 10.2.214.51
in-destination 10.2.214.63
out-realm-id access
out-source 10.1.215.63
out-destination 0.0.0.0
protocol ICMP
alg-type none
start-port 0
end-port 0
flow-time-limit 0
initial-guard-timer 60
subsq-guard-timer 60
average-rate-limit 0
```

15

High Availability Nodes

Oracle USM s can be deployed in pairs to deliver high availability (HA). Two Oracle USM s operating in this way are called an HA node. Over the HA node, media and call state are shared, keeping sessions/calls from being dropped in the event of a failure.

Two Oracle USM s work together in an HA node, one in active mode and one in standby mode.

- The active Oracle USM checks itself for internal process and IP connectivity issues. If it detects that it is experiencing certain faults, it will hand over its role as the active system to the standby Oracle USM in the node.
- The standby Oracle USM is the backup system, fully synchronized with active Oracle USM s session status. The standby Oracle USM monitors the status of the active system so that, if needed, it can assume the active role without the active system having to instruct it to do so. If the standby system takes over the active role, it notifies network management using an SNMP trap.

In addition to providing instructions for how to configure HA nodes and their features, this chapter explains how to configure special parameters to support HA for all protocols.

Overview

To produce seamless switchovers from one Oracle USM to the other, the HA node uses shared virtual MAC and virtual IP addresses for the media interfaces in a way that is similar to VRRP (virtual router redundancy protocol). When there is a switchover, the standby Oracle USM sends out a gratuitous ARP messages using the virtual MAC address, establishing that MAC on another physical port within the Ethernet switch. To the upstream router, the MAC and IP are still alive, meaning that existing sessions continue uninterrupted.

Within the HA node, the Oracle USM s advertise their current state and health to one another in checkpointing messages; each system is apprised of the other's status. Using Oracles HA protocol, the Oracle USM s communicate with UDP messages sent out and received on the rear interfaces.

The standby Oracle USM shares virtual MAC and IPv4 addresses for the media interfaces (similar to VRRP) with the active Oracle USM . Sharing addresses eliminates the possibility that the MAC and IPv4 address set on one Oracle USM in an HA node will be a single point of failure. The standby Oracle USM sends ARP requests using a utility IPv4 address and its hard-coded MAC addresses to obtain Layer 2 bindings.

The standby Oracle USM assumes the active role when:

- It has not received a checkpoint message from the active Oracle USM for a certain period of time.
- It determines that the active Oracle USM 's health score has decreased to an unacceptable level.
- The active Oracle USM relinquishes the active role.

Establishing Active and Standby Roles

Oracle USM s establish active and standby roles in the following ways.

- If a Oracle USM boots up and is alone in the network, it is automatically the active system. If you then pair a second Oracle USM with the first to form an HA node, then the second system to boot up will establish itself as the standby automatically.
- If both Oracle USM s in the HA node boot up at the same time, they negotiate with each other for the active role. If both systems have perfect health, then the Oracle USM with the lowest HA rear interface IPv4 address will become the active Oracle USM . The Oracle USM with the higher HA rear interface IPv4 address will become the standby Oracle USM .
- If the rear physical link between the two Oracle USM s fails during boot up or operation, both will attempt to become the active Oracle USM . In this case, processing will not work properly.

Health Score

HA Nodes use health scores to determine their active and standby status. Health scores are based on a 100-point system. When a Oracle USM is functioning properly, its health score is 100.

Generally, the Oracle USM with the higher health score is active, and the Oracle USM with the lower health score is standby. However, the fact that you can configure health score thresholds builds some flexibility into using health scores to determine active and standby roles. This could mean, for example, that the active Oracle USM might have a health score lower than that of the standby Oracle USM , but a switchover will not take place because the active Oracle USM 's health score is still above the threshold you configured.

Alarms are key in determining health score. Some alarms have specific health score value that are subtracted from the Oracle USM 's health score when they occur. When alarms are cleared, the value is added back to the Oracle USM 's health score.

You can look at a Oracle USM 's health score using the ACLI **show health** command.

Switchovers

A switchover occurs when the active Oracle USM stops being the active system, and the standby system takes over that function. There are two kinds switchovers: automatic and manual.

Automatic Switchovers

Automatic switchovers are triggered without immediate intervention on your part. Oracle USM s switch over automatically in the following circumstances:

- When the active Oracle USM 's health score of drops below the threshold you configure.
- When a time-out occurs, meaning that the active Oracle USM has not has not sent checkpointing messages to the standby Oracle USM within the allotted time.
  The active Oracle USM might not send checkpointing messages for various reasons such as link failure, communication loss, or advertisement loss. Even if the active Oracle USM has a perfect health score, it will give up the active role if it does not send a checkpoint

message or otherwise advertise its status within the time-out window. Then the standby Oracle USM takes over as the active system.

When an automatic switchover happens, the Oracle USM that has just become active sends an ARP message to the switch. This message informs the switch to send future messages to its MAC address. The Oracle USM that has just become standby ignores any messages sent to it.

Manual Switchovers

You can trigger a manual switchover in the HA node by using the ACLI **notify berpd force** command. This command forces the two Oracle USM s in the HA node to trade roles. The active system becomes standby, and the standby becomes active.

In order to perform a successful manual switchover, the following conditions must be met.

- The Oracle USM from which you trigger the switchover must be in one of the following states: active, standby, or becoming standby.
- A manual switchover to the active state is only allowed on a Oracle USM in the standby or becoming standby state if it has achieved full media, signaling, and configuration synchronization.
- A manual switchover to the active state is only allowed on a Oracle USM in the standby or becoming standby state if it has a health score above the value you configure for the threshold.

State Transitions

Oracle USM s can experience series of states as they become active or become standby.

Note:

Packet processing only occurs on an active Oracle USM .

State	Description
Initial	When the Oracle USM is booting.
Becoming Active	When the Oracle USM has negotiated to become the active system, but is waiting the time that you set to become fully active. Packets cannot be processed in this state.
Active	When the Oracle USM is handling all media, signaling, and configuration processing.
Relinquishing Active	When the Oracle USM is giving up its Active status, but before it has become standby. This state is very brief.
Becoming Standby	When the Oracle USM is becoming the standby system but is waiting to become fully synchronized. It remains in this state for the period of time you set in the becoming-standby-time parameter, or until it is fully synchronized.
Standby	When the Oracle USM is fully synchronized with its active system in the HA node.
OutOfService	When the Oracle USM cannot become synchronized in the period of time you set in the becoming-standby-time parameter.

State Transition Sequences

When the active Oracle USM assumes its role as the as the active system, but then changes roles with the standby Oracle USM to become standby, it goes through the following sequence of state transitions:

- Active
- RelinquishingActive
- BecomingStandby
- Standby
  When the standby Oracle USM assumes its role as the standby system, but then changes roles with the active Oracle USM to become active, it goes through the following sequence of state transitions:
- Standby
- BecomingActive
- Active

HA Features

HA nodes support configuration checkpointing, which you are required to set up so that the configurations across the HA node are synchronized. In addition, you can set up the following optional HA node features:

- Multiple rear interface support
- Gateway link failure detection and polling

Multiple Rear Interfaces

Configuring your HA node to support multiple rear interfaces eliminates the possibility that either of the rear interfaces you configure for HA support will become a single point of failure. Using this feature, you can configure individual Oracle USM s with multiple destinations on the two rear interfaces, creating an added layer of failover support.

When you configure your HA node for multiple rear interface support, you can use last two rear interfaces (wancom1 and wancom2) for HA—the first (wancom0) being used for Oracle USM management. You can connect your Oracle USM s using any combination of wancom1 and wancom2 on both systems. Over these rear interfaces, the Oracle USM s in the HA node share the following information:

- Health
- Media flow
- Signaling
- Configuration

For example, if one of the rear interface cables is disconnected or if the interface connection fails for some other reason, all health, media flow, signaling, and configuration information can be checkpointed over the other interface.

Health information is checkpointed across all configured interfaces. However, media flow, signaling, and configuration information is checkpointed across one interface at a time, as determined by the Oracle USM 's system HA processes.

Configuration Checkpointing

During configuration checkpointing, all configuration activity and changes on one Oracle USM are automatically mirrored on the other. Checkpointed transactions include adding, deleting, or modifying a configuration on the active Oracle USM . This means that you only need to perform configuration tasks on the active Oracle USM because the standby system will go through the checkpointing process and synchronize its configuration to reflect activity and changes.

Because of the way configuration checkpointing works, the ACLI **save-config** and **activate-config** commands can only be used on the active Oracle USM .

- When you use the ACLI **save-config** command on the active Oracle USM , the standby Oracle USM learns of the action and updates its own configuration. Then the standby Oracle USM saves the configuration automatically.
- When you use the ACLI **activate-config** command on the active Oracle USM , the standby Oracle USM learns of the action and activates its own, updated configuration.

The ACLI **acquire-config** command is used to copy configuration information from one Oracle USM to another.

Gateway Link Failure Detection and Polling

In an HA node, the Oracle USM s can poll for and detect media interface links to the gateways as they monitor ARP connectivity. The front gateway is assigned in the network interface configuration, and is where packets are forwarded out of the originator's LAN.

The Oracle USM monitors connectivity using ARP messages that it exchanges with the gateway. The Oracle USM sends regular ARP messages to the gateway in order to show that it is still in service; this is referred to as a heartbeat message. If the Oracle USM deems the gateway unreachable for any of the reasons discussed in this section, a network-level alarm is generated and an amount you configure for this fault is subtracted from the system's health score.

TheOracle USM generates a gateway unreachable network-level alarm if the Oracle USM has not received a message from the media interface gateway within the time you configure for a heartbeat timeout. In this case, The Oracle USM will send out ARP requests and wait for a reply. If no reply is received after resending the set number of ARP requests, the alarm remains until you clear it. The health score also stays at its reduced amount until you clear the alarm.

When valid ARP requests are once again received, the alarm is cleared and system health scores are increased the appropriate amount.

You can configure media interface detection and polling either on a global basis in the SD HA nodes/redundancy configuration or on individual basis for each network interface in the network interface configuration.

> **Note:**
>
> To improve the detection of link failures, the switchport connected to the NIU should have Spanning Tree disabled. Enabling Spanning Tree stops the switchport from forwarding frames for several seconds after a reset. This prevents the NIU from reaching the gateway and generates a "gateway unreachable" network-level alarm.

Before Configuring a High Availability (HA) Pair

> **Note:**
>
> When you configure an HA pair, you must use the same password for both Oracle USMs.
> Before configuring the parameters that support HA, complete the following steps.

1. Establish the physical connections between the Oracle USMs. Avoid breaking the physical link (over the rear interfaces) between the Oracle USMs in an HA node. If the physical link between the Oracle USMs breaks, they will both attempt to become the active system and HA will not function as designed.
2. Confirm that both Oracle USMs are set to the same time. Use the ACLI **show clock** command to view the system time. If the Oracle USMs show different times, use the **system-timeset** command to change the time.

   Oracle recommends that you use NTP to synchronize your Oracle USMs, so that they have a common stratum time source.
3. HA nodes use ports 1 and 2 as the HA interfaces. Set port 0 on the rear panel of the Oracle USM chassis as the boot and management interface. You configure the rear interfaces during the physical interface configuration.
4. For ACLI configuration, you need to know the target names of the Oracle USMs making up the HA node. The target name of the system is reflected in the ACLI's system prompt. For example, in the ORACLE# system prompt, ORACLE is the target name.

   You can also see and set the target name in the Oracle USM boot parameters.

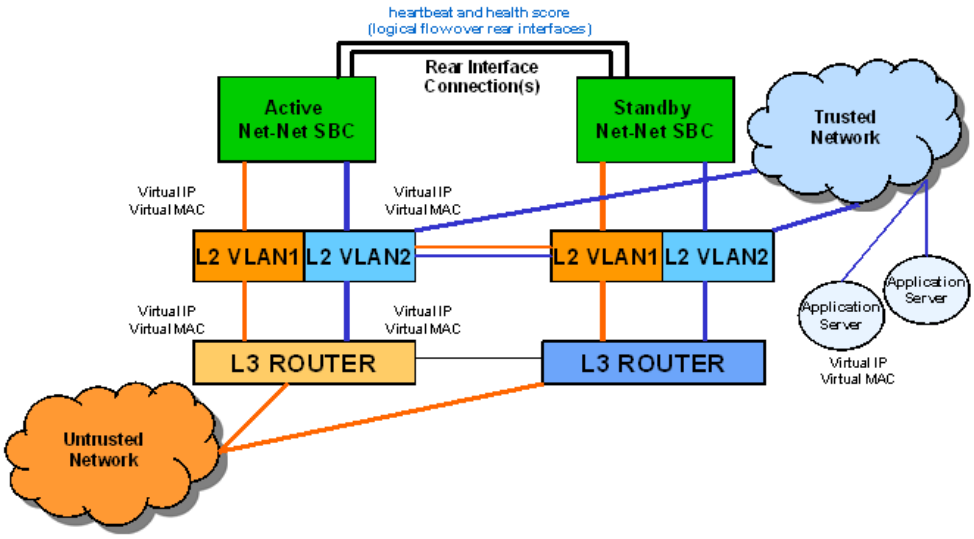
   > **Note:**
   >
   > The target name is case sensitive.
5. Devise virtual MAC addresses so that, if a switchover happens, existing sessions are not interrupted.

HA Node Connections

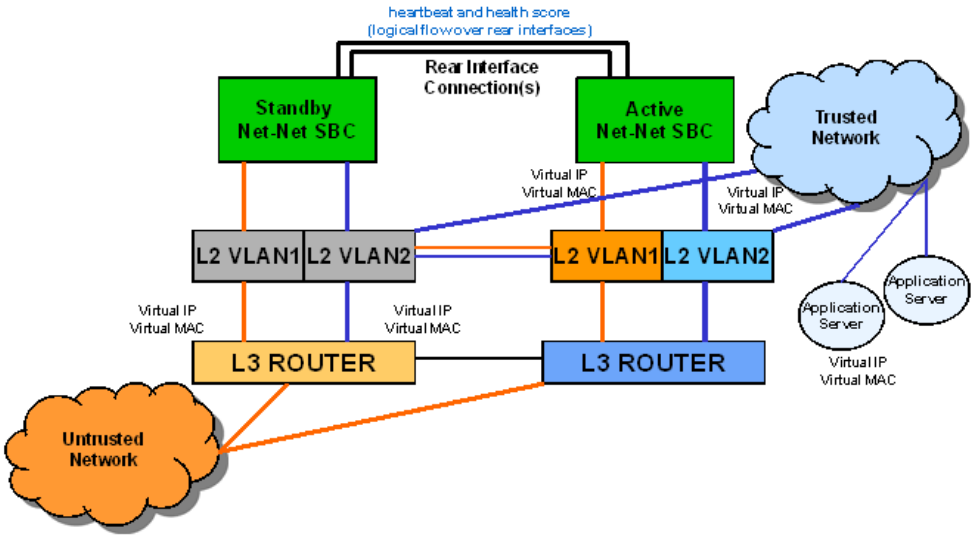
To use HA, you must establish Layer 2 and Layer 3 networks that interconnect two Oracle USMs and support HA with the required physical network connections. The basic network set-up in the following diagram shows an HA node deployment where each system is connected to

its own Layer 2 switch. This set-up provides a measure of added redundancy in the event that one of the switches fails.

Here, the active system is using the virtual MAC and IP addresses.



In the second diagram, the same network is shown with the HA node having experienced a switchover. The previously standby Oracle USM has taken over the active role in the HA node and is using the virtual IP and MAC addresses.



> **Note:**
>
> Switches should never be in master-slave mode. If they are, HA will not work correctly.

The following are hardware set-up and location considerations for placing an HA Node:

- You must set up each Oracle USM according to the requirements and safety precautions set out in the *Oracle Communications System Hardware Installation Guide*.
- Each Oracle USM's media interfaces must be connected to the same switches (or other network entities), as shown in the diagram above.
- The length of the shielded crossover 10/100 category 5 Ethernet cable that connects the Oracle USMs from the rear interfaces must be able to reach from the configured rear interface on one Oracle USM to the configured rear interface on the other.

HA nodes use Oraclerder element redundancy protocol for its tasks. This protocol uses a connection between the rear interfaces of two Oracle USMs to checkpoint the following information: health, state, media flow, signaling, and configuration.

We recommend that you use shielded category 5 (RJ45) crossover cables for all 10/100 Ethernet connections used for HA.

You can set up either single or multiple rear interface support for your HA node. For single interface support, one cable connects the two Oracle USMs; for multiple interface support, two cables are used. However, the software configurations for each type of connection mode are different.

When you make these connections, do not use port 0 (wancom0) on the rear interface of the Oracle USM chassis; that port should only be used for Oracle USM management. Instead, use ports 1 and 2 (wancom1 and wancom2).

To cable Oracle USMs using single rear interface support:

1. Using a 10/100 category 5 crossover cable, insert one end into either port 1 (wancom1) or port 2 (wancom2) on the rear interface of the first Oracle USM.
2. Insert the other end of the cable into port 1 or port 2 on the rear interface of the second Oracle USM. We recommend that you use corresponding ports on the two systems. That is, use port 1 on both systems or use port 2 on both systems.
3. Perform software configuration for these interfaces as described in this chapter.

   To cable Oracle USMs using multiple rear interface support:

4. Using a 10/100 category 5 crossover cable, insert one end into port 1 on the rear interface of the first Oracle USM.
5. Insert the other end of that cable into port 1 on the rear interface of the second Oracle USM to complete the first physical connection.
6. Using a second 10/100 category 5 cable, insert one end into port 2 on the rear interface of the first Oracle USM.
7. Insert the other end of this second cable in port 2 on the rear interface of the second Oracle USM to complete the second physical connection.
8. Perform software configuration for these interfaces as described in this chapter.

Virtual MAC Addresses

In order to create the HA node, you need to create virtual MAC addresses for the media interfaces. You enter these addresses in virtual MAC address parameters for physical interface configurations where the operation type for the interface is media.

The HA node uses shared virtual MAC (media access control) and virtual IP addresses for the media interfaces. When there is a switchover, the standby Oracle USM sends out an ARP message using the virtual MAC address, establishing that MAC on another physical port within the Ethernet switch. Virtual MAC addresses are actually unused MAC addresses that based on the Oracle USM's root MAC address.

The MAC address is a hardware address that uniquely identifies each Oracle USM. Given that, the virtual MAC address you configure allows the HA node to appear as a single system from the perspective of other network devices. To the upstream router, the MAC and IP are still alive, meaning that existing sessions continue uninterrupted through the standby Oracle USM.

Depending on the type of physical layer cards you have installed, you can create MAC addresses as follows: Four Ethernet (MAC) address for each configured four-port GigE physical interface card.

Virtual MAC Address Configuration

To create a virtual MAC address:

1. Determine the Ethernet address of the Oracle USM by using the ACLI **show interfaces** command. This command only works if you have already set up physical interface configurations. Otherwise, you will get no output.

The example below shows you where the Ethernet address information appears; this sample has been shortened for the sake of brevity. For each type of physical interface card, the Oracle USM displays the following:

```
ORACLE# show interfaces
f00 (media slot 0, port 0)
     Flags: UP BROADCAST MULTICAST ARP RUNNING
     Type: GIGABIT_ETHERNET
     Admin State: enabled
     Auto Negotiation: enabled
     Internet address: 10.10.0.10     Vlan: 0
     Broadcast Address: 10.10.255.255
     Netmask: 0xffff0000
     Gateway: 10.10.0.1
     Ethernet address is 00:08:25:01:07:64
```

2. Identify the root portion of the Ethernet (MAC) address.

   Each Oracle USM has MAC addresses assigned to it according to the following format: 00:08:25:XX:YY:ZN where:

   - 00:08:25 refers to Acme Packet
   - XX:YY:ZN refers to the specific Oracle USM
   - N is a 0-f hexadecimal value available for the Oracle USM

     In this example, the root part of this address is 00:08:25:XX:YY:Z.

3. To create an unused MAC address (that you will use as the virtual MAC address) take the root MAC address you have just identified. Replace this N value with unused hexadecimal values for the Oracle USM: 8, 9, e, or f.

   In other words, you change the last digit of the MAC address to either 8, 9, e, or f depending on which of those address are not being used.

   For example, for an HA node with MAC address bases of 00:08:25:00:00:00 and 00:08:25:00:00:10, the following addresses would be available for use at virtual MAC addresses:

   - 00:08:25:00:00:08
   - 00:08:25:00:00:09
   - 00:08:25:00:00:0e
   - 00:08:25:00:00:0f
   - 00:08:25:00:00:18
   - 00:08:25:00:00:19
   - 00:08:25:00:00:1e
   - 00:08:25:00:00:1f

     Corresponding media interfaces in HA nodes must have the same virtual MAC addresses. Given that you have various physical interface card options, the following points illustrate how virtual MAC address can be shared:

   If you are using a four-port GigE physical interface card, both the active Oracle USMand the standby Oracle USM might have the following virtual MAC address scheme for the slots:

   - Slot 0 _ 00:08:25:00:00:0e and 00:08:25:00:00:0f
   - Slot 1 - 00:08:25:00:00:1e and 00:08:25:00:00:1f

> **Note:**
>
> Note the virtual MAC addresses you have created so that you can reference them easily when you are configuring the physical interfaces for HA.

HA Node Connections

You can begin software configuration for your HA node after you have:

- Completed the steps for physical set-up and connection.
- Noted the target name of the Oracle USMs that make up the HA node.
- Configured the virtual MAC addresses that you need, according to the type of physical interface cards installed on your Oracle USM.

HA Node Connection Configuration

If you are using HA, you need to set the physical interface configuration parameters described in this section to establish successful connections. These parameters are for rear and media interfaces.

To access physical interface menu in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **phy-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# phy-interface
   ORACLE(phy-interface)#
   ```

   From this point, you can configure physical interface parameters. To view all physical interfaces parameters, enter a **?** at the system prompt.

Rear Interfaces

You can use port 1 (wancom1) or port 2 (wancom2) as interfaces to support HA. Do not use port 0 (wancom 0) as that port is reserved for carrying management traffic.

Make sure that the physical connections you have made on the rear panel of your Oracle USMs correspond to the configurations you enter for physical interfaces. You can connect Oracle USMs through multiple rear interfaces. For multiple rear interface connectivity, cable both port 1 and port 2 (wancom1 and wancom2) on one Oracle USM to port1 and port 2 on the other Oracle USM in the HA node.

The Oracle USM's HA function depends heavily on health scores to determine the active and standby roles in an HA node. You can set the amount that will be subtracted from a Oracle USM's health score in the event that a management interface fails for any reason. For example, a connection might become invalid or a cable might be removed inadvertently.

The following example shows how a configured physical interface will appear in the ACLI for an HA node:

```
phy-interface
        name                            wancom1
        operation-type                  Control
        port                            1
        slot                            0
        virtual-mac
        wancom-health-score             20
```

To establish rear interfaces for use in an HA node using the ACLI:

1. Access the physical interface menu.
2. **name**—Set a name for the interface using any combination of characters entered without spaces. For example: wancom1.
3. **operation-type**—Set this parameter to **Control**.
4. **slot**—Set this parameter to **0**.
5. **port**—Set this parameter to **1** or **2**.
6. **wancom-health-score**—Enter the number value between 0 and 100. This value will be subtracted from the Oracle USM's health score in the event that a rear interface link fails. We recommend that you change this value from its default (**50**), and set it to **20**.

   This value you set here is compared to the active and emergency health score thresholds you establish in the Oracle USM HA node (redundancy) configuration.
7. For multiple rear interface support, configure the remaining, unused rear interfaces with the appropriate values.

   The following example shows configuration for multiple rear interface support.

```
ORACLE(system)# phy-interface
ORACLE(phy-interface)# name wancom1
ORACLE(phy-interface)# operation-type control
ORACLE(phy-interface)# port 1
ORACLE(phy-interface)# wancom-health-score 20
ORACLE(phy-interface)# done
ORACLE(phy-interface)# name wancom2
ORACLE(phy-interface)# operation-type control
ORACLE(phy-interface)# port 2
ORACLE(phy-interface)# wancom-health-score 20
ORACLE(phy-interface)# done
```

Media Interface Virtual MAC Addresses

To configure HA for the media interfaces in an HA node, you must set one or more virtual MAC addresses, according to the type of physical layer cards you have installed on your Oracle USM.

To set a virtual MAC address using the ACLI:

1. Access the physical interface configuration.
2. Configure all relevant parameters as noted in the Physical Interfaces section of this guide's *System Configuration* chapter.

   Since virtual MAC addresses are used for media interfaces only, verify that the operation type is set to media.

3. **virtual-mac**—Enter the virtual MAC address that you have created using the steps in the Virtual MAC Addresses section.

HA Node Parameters

To establish a pair of Oracle USMs as an HA node, you need to configure basic parameters that govern how the Oracle USMs:

- Transition on switchover
- Share media and call state information
- Checkpoint configuration data

The following example shows what an HA configuration might look like in the ACLI.

```
redundancy-config
        state                           enabled
        log-level                       WARNING
        health-threshold                75
        emergency-threshold             50
        port                            9090
        advertisement-time              500
        percent-drift                   210
        initial-time                    1250
        becoming-standby-time           45000
        becoming-active-time            100
```

You need to configure the two Oracle USMs to be HA node peers. To enable configuration checkpointing, you must to configure two peers in the ACLI, one for the primary and one for the secondary Oracle USM. The HA node peers configuration also allows you to configure destinations for where to send health and state information. Unless you create Oracle USM peers and destinations configurations, HA will not work properly.

The following example shows what an HA configuration might look like in the ACLI.

```
peer
                name                            netnetsd1
                state                           enabled
                type                            Primary
                destination
                        address                         169.254.1.1:9090
network-interface               wancom1:0
peer
                name                            netnetsd2
                state                           enabled
                type                            Secondary
                destination
                        address                         169.254.1.2:9090
                        network-interface               wancom1:0
```

HA Node Parameter Configuration

To configure general HA node parameters using the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **redundancy** and press Enter.

   ```
   ORACLE(system)# redundancy
   ```

   From here, you configure basic HA node parameters. To view all basic HA node parameters, enter a **?** at the system prompt.

4. **state**—Leave this parameter set to **enabled** for HA to work. To stop HA operation, set this parameter to **disabled**. The default value is **enabled**. The valid values are:

   - enabled | disabled

5. **log-level**—Set the log level you want to use for the HA system process. The value you set in this field overrides any log level value you set for the entire Oracle USM in the system configuration process log level parameter. The default value is **INFO** which allows you to receive a moderate amount of detail. The valid values are:

   - emergency | critical | major | minor | warning | notice | info | trace | debug | detail

6. **health-threshold**—Enter a value between 0 and 100 to set the health score at which the Oracle USMs in the HA node gracefully exchange active-standby roles. The default value is **75**. The valid range is:

   - Minimum—1
   - Maximum—100

     For example, if this field is set to **75** and the active Oracle USM's health score falls below that point, the standby Oracle USM will take over the active role. However, Oracle USM will only take over the active role if its own health score is 75 or better.

7. **emergency-threshold**—Enter the health score for the standby Oracle USM to become active immediately. The default value is **50**. The valid range is:

   - Minimum—0
   - Maximum—100

     If the standby Oracle USM is initializing and the active Oracle USM's health score is below the health threshold, the standby Oracle USM will take the active role and there will be a graceful switchover. If the active Oracle USM's health score is below the emergency threshold, then the switchover will be immediate.

     If the standby Oracle USM has a health score below the emergency threshold and the active Oracle USMis unhealthy, the active Oracle USM will not give up its active role.

8. **advertisement-time**—Enter the number of milliseconds to set how often Oracle USMs in an HA node inform each other of their health scores.

   We recommend you leave this parameter set to it's default, **500**. The valid range is:

   - Minimum—50
   - Maximum—999999999

9. **percent-drift**—Enter the percentage of the advertisement time that you want one member of the HA node to wait before considering the other member to be out of service. For the standby Oracle USM, this is the time it will wait before taking the active role in the HA node. The default value is **210**. The valid range is:

   - Minimum—100
   - Maximum—65535

10. **initial-time**—Enter the number of milliseconds to set the longest amount of time the Oracle USM will wait at boot time to change its state from initial to either becoming active or becoming standby. The default value is **1250**. The valid range is:
    - Minimum—5
    - Maximum—999999999
11. **becoming-standby-time**—Enter the number of milliseconds the Oracle USM waits before becoming standby, allowing time for synchronization. If it is not fully synchronized within this time, it will be declared out of service.

    We recommend that you do not set this parameter below **45000**. If a large configuration is being processed, we recommend setting this parameter to **180000** to allow enough time for configuration checkpointing. The default value is **45000**. The valid range is:
    - Minimum—5
    - Maximum—999999999
12. **becoming-active-time**—Enter the number of milliseconds that the standby Oracle USM takes to become active in the event that the active Oracle USM fails or has an intolerably decreased health score. The default value is **100**. The valid range is:
    - Minimum—5
    - Maximum—999999999

HA Node Peer Configuration

To configure a Oracle USM as an HA node peer:

1. From the redundancy menu, type **peers** and press Enter.

   ```
   ORACLE(system)# redundancy
   ORACLE(redundancy)# peers
   ```
2. **state**—Enable or disable HA for this Oracle USM. The default value is **enabled**. The valid values are:
   - enabled | disabled
3. **name**—Set the name of the HA node peer as it appears in the target name boot parameter.

   This is also the name of your system that appears in the system prompt. For example, in the system prompt ORACLE1#, ORACLE1 is the target name for that Oracle USM.
4. **type**—These values refer to the primary and secondary utility addresses in the network interface configuration. To determine what utility address to use for configuration checkpointing, set the type of Oracle USM: primary or secondary.

   > **Note:**
   >
   > You must change this field from unknown, its default. The valid values are:

   - **primary**—Set this type if you want the Oracle USM to use the primary utility address.
   - **secondary**—Set this type if you want the Oracle USM to use the secondary utility address.
   - **unknown**—If you leave this parameter set to this default value, configuration checkpointing will not work.

HA Node Health And State Configuration

To configure where to send health and state information within an HA node:

1. From the peers configuration, type destinations and press Enter.

   ```
   ORACLE(rdncy-peer)# destinations
   ORACLE(rdncy-peer-dest)#
   ```

2. **address**—Set the destination IPv4 address and port of the other Oracle USM in the HA node to which this Oracle USM will send HA-related messages. This value is an IPv4 address and port combination that you enter as: IPAddress:Port. For example, 169.254.1.1:9090.

   - The IPv4 address portion of this value is the same as the IPv4 address parameter set in a network interface configuration of the other Oracle USM in the HA node.
   - The port portion of this value is the port you set in the Oracle USM HA Node/ redundancy configuration for the other Oracle USM in the node.

3. **network-interface**—Set the name and subport for the network interface where the Oracle USM receives HA-related messages. Valid names are wancom1 and wancom2. This name and subport combination must be entered as name:subport; for example, **wancom1:0**.

   The network interface specified in this parameter must be linked to a physical interface configured with rear interface parameters. The physical interface's operation type must be control or maintenance, and so the subport ID portion of this parameter is 0. The subport ID is the VLAN tag.

Synchronizing Configurations

You can synchronize the Oracle USMs (USM) in your High Availability (HA) node in the following ways:

- Automatically — Set up configuration checkpointing within the HA node.
- Manually — Check whether or not configurations in the HA node are synchronized, and then copy configuration data from one USM to the other.

When you initially configure a new HA node, copy the configuration data manually from one USM to the other. When you complete the process, you can configure your HA node to automatically synchronize configurations.

Oracle recommends that you configure the HA node for configuration checkpointing because that is the most reliable way to ensure that both systems have the same configuration.

Synchronize HA Peers

The process for synchronizing the peers in a High Availability (HA) node for the first time by way of the ACLI includes the following steps.

1. Create a complete configuration on the active Oracle USM (USM). Include all HA node parameters and all rear interface configurations. Confirm that the rear interfaces are configured to send and receive information across the HA node.
2. On the active USM, save the configuration.
3. On the active USM, reboot to run the new configuration.

Use the ACLI **show health** command to see that the active USM booted without a peer. This changes after you copy the configuration to the standby USM and activate the configuration.

4. On the standby USM, perform the ACLI **acquire-config** command to copy the configuration from the active USM. Use the **acquire-config** command with the IPv4 address of wancom 0 on the active USM.

   ```
   ACMEPACKET2# acquire-config 192.168.12.4
   ```

   The IPv4 address of wancom 0 on the active USM is the IPv4 address portion of the value displayed for the inet on ethernet boot parameter. The following codeblock shows an example of the inet on ethernet value that the system displays when you view the boot parameters:

   ```
   inet on ethernet (e)    : 192.168.12.4:ffff0000
   ```

5. When the copying process (**acquire-config**) is complete, reboot the standby USM to activate the configuration. The system boots and displays start-up information.

6. Confirm that the HA node synchronized the configurations by using the ACLI **display-current-cfg-version** and **display-running-cfg-version** commands:

   ```
   ORACLE# display-current-cfg-version
   Current configuration version is 3
   ORACLE# display-running-cfg-version
   Running configuration version is 3
   ORACLE# display-current-cfg-version
   Current configuration version is 3
   ORACLE# display-running-cfg-version
   Running configuration version is 3
   ```

   In the preceding example, all configuration versions—current and running—are the same number (3).

Using Configuration Checkpointing

The Oracle USM's primary and secondary utility addresses support configuration checkpointing, allowing the standby Oracle USM to learn configuration changes from the active Oracle USM. This means that you only have to enter configuration changes on the active Oracle USM for the configurations across the HA node to be updated.

Configuration checkpointing uses parameters in the network interface and in the Oracle USM HA Nodes/redundancy configurations.

If you are using configuration checkpointing, you also need to set up two Oracle USM peer configurations: one the primary, and one for the secondary.

HA Configuration Checkpointing

You need to first set applicable network interface configuration parameters, and then establish applicable parameters in the Oracle USM HA node (redundancy) configuration.

We recommend that you do not change the configuration checkpointing parameters in the redundancy configuration. Using the defaults, this feature will function as designed.

> **Note:**
>
> Remember to set the appropriate type parameter in the HA node redundancy peers configuration.

For the network interface, these parameters appear as they do in the following example when you use the ACLI. This example has been shortened for the sake of brevity.

```
pri-utility-addr              169.254.1.1
sec-utility-addr              169.254.1.2
```

For the Oracle USM HA node (redundancy) configuration, these parameters appear as they do in the following example when you use the ACLI. This example has been shortened for the sake of brevity. You should not change these values without consultation from Oracle Technical Support or your Oracle Systems Engineer.

```
cfg-port                      1987
cfg-max-trans                 10000
cfg-sync-start-time           5000
cfg-sync-comp-time            1000
```

To configure HA configuration checkpointing in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **network-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(system)# network-interface
   ORACLE(network-interface)#
   ```

   From here, you can configure network interface parameters. To view all network interfaces parameters, enter a **?** at the system prompt.

4. **pri-utility-addr**—Enter the utility IP address for the primary HA peer in an HA architecture.

   This address can be any unused IP address within the subnet defined for the network interface. For example, given a network interface of with the IPv4 address 168.0.4.15/24 (identifying the host associated with the network interface), the possible range of unused IPv4 addresses is 168.0.4.1 to 168.0.4.254. Your network administrator will know which IPv4 addresses are available for use.

5. **sec-utility-addr**—Enter the utility IP address for the secondary Oracle USM peer in an HA architecture.

   Usually, this IP address is usually the next in the sequence up from the primary utility address. It is also generated from the range of unused IP addresses within the subnet defined for the network interface.

6. Save your work and exit the network interface configuration.

   ```
   ORACLE(network-interface)# done
   ORACLE(network-interface)# exit
   ORACLE(system)#
   ```

7. Access the system HA node/redundancy configuration by typing **redundancy** at the system prompt and then press Enter.

```
ORACLE(system)# redundancy
ORACLE(redundancy)#
```

> **Note:**
>
> We strongly recommend that you keep the default settings for the parameters Steps 8 through 11.

8. **cfg-port**—Enter the port number for sending and receiving configuration checkpointing messages. Setting this to zero (**0**) disables configuration checkpointing. The default value is **1987**. The valid values are:
   - Minimum—0, 1025
   - Maximum—65535
9. **cfg-max-trans**—Enter the number of HA configuration checkpointing transactions that you want to store. The active Oracle USM maintains the transaction list, which is acquired by the standby Oracle USM. Then the standby system uses the list to synchronize its configuration with active system. The default value is **10000**. The valid range is:
   - Minimum—0
   - Maximum—4294967295

     Transactions include: modifications, additions, and deletions. If the maximum number of stored transactions is reached, the oldest transactions will be deleted as new transactions are added.
10. **cfg-sync-start-time**—Enter the number of milliseconds before the Oracle USM tries to synchronize by using configuration checkpointing. On the active Oracle USM, this timer is continually reset as the Oracle USM checks to see that it is still in the active role. If it becomes standby, it waits this amount of time before it tries to synchronize.

    We recommend you leave this field at its default value, **5000**, so that configuration checkpointing can function correctly. The valid range is:
    - Minimum—0
    - Maximum—4294967295
11. **cfg-sync-comp-time**—Enter the number of milliseconds that the standby Oracle USM waits before checkpointing to obtain configuration transaction information after the initial checkpointing process is complete.

    We recommend you leave this field at its default value, **1000**, so that configuration checkpointing can function correctly. The valid range is:
    - Minimum—0
    - Maximum—4294967295
12. Save your work and exit the redundancy configuration.

```
ORACLE(redundancy)# done
ORACLE(redundancy)# exit
ORACLE(system)#
```

Manually Checking Configuration Synchronization

You can check that the current and active configurations are synchronized across the HA node. The current configuration is the one with which you are currently working, and the active configuration is the one active on the system.

To confirm that the systems in the HA node have synchronized configurations:

1. On the active Oracle USM in the Superuser menu, enter the following ALCI commands and press Enter. Note the configuration version numbers for comparison with those on the standby Oracle USM.

   - **display-current-cfg-version**—Shows the version number of the configuration you are currently viewing (for editing, updating, etc.).

     ```
     ORACLE# display-current-cfg-version
     Current configuration version is 30
     ```

   - **display-running-cfg-version**—Shows the version number of the active configuration running on the Oracle USM.

     ```
     ORACLE# display-running-cfg-version
     Running configuration version is 30
     ```

2. On the standby Oracle USM, enter the following ALCI commands and press Enter. Note the configuration version numbers for comparison with those on the active Oracle USM.

   ```
   ORACLE# display-current-cfg-version
   Current configuration version is 30
   ORACLE# display-running-cfg-version
   Running configuration version is 30
   ```

3. Compare the configuration numbers. If the version numbers on the active Oracle USM match those on the standby Oracle USM, then the systems are synchronized.

   If the version numbers do not match, you need to synchronize the Oracle USMs. You can do so using the ACLI **acquire-config** command.

Media Interface Link Detection and Gateway Polling

You can use media interface link detection and gateway polling globally on the Oracle USM, or you can override those global parameters on a per-network-interface basis.

- Use the Oracle USM HA node (redundancy) configuration to establish global parameters. When configured globally, they will appear like this in the ACLI:

  ```
  gateway-heartbeat-interval     0
  gateway-heartbeat-retry        0
  gateway-heartbeat-timeout      1
  gateway-heartbeat-health       0
  ```

- Use the network interface's gateway heartbeat configuration to override global parameters on a per-network-interface basis. When configured for the network interface, these parameters will appear like this in the ACLI:

  ```
  gw-heartbeat
                state                          enabled
                heartbeat                      0
                retry-count                    0
                retry-timeout                  1
                health-score                   0
  ```

Media Interface Link Detection and Gateway Polling Configuration

To configure global media interface link detection and gateway polling:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# system
   ```

3. Type **redundancy** and press Enter.

   ```
   ORACLE(system)# redundancy
   ```

   From here, you can configure gateway heartbeat parameters. To view all gateway heartbeat parameters, enter a **?** at the system prompt.

4. **gateway-heartbeat-interval**—Enter the number of seconds between heartbeats for the media interface gateway. Heartbeats are sent at this interval as long as the media interface is viable. The default value is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

5. **gateway-heartbeat-retry**—Enter the number of heartbeat retries (subsequent ARP requests) to send to the media interface gateway before it is considered unreachable. The default value is **0**. The valid range is:

   - Minimum—0
   - Maximum—65535

6. **gateway-heartbeat-timeout**—Enter the heartbeat retry time-out value in seconds. The default value is **1**. The valid range is:

   - Minimum—0
   - Maximum—65535

     This parameter sets the amount of time between Oracle USM ARP requests to establish media interface gateway communication after a media interface gateway failure.

7. **gateway-heartbeat-health**—Enter the amount to subtract from the Oracle USM's health score if a media interface gateway heartbeat fails. If the value you set in the gateway time-out retry field is exceeded, this amount will be subtracted from the system's overall health score. The default value is **0**. The valid range is:

   - Minimum—0
   - Maximum—100

Media Interface Link Detection and Gateway Polling Configuration 2

To configure media interface link detection and gateway polling on a per-network-interface basis in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **system** and press Enter to access the system-level configuration elements.

   ```
   ACMEPACKET(configure)# system
   ```

3. Type **network-interface** and press Enter.

   ```
   ACMEPACKET(system)# network-interface
   ```

4. Type **gw-heartbeat** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(network-interface)# gw-heartbeat
   ACMEPACKET(gw-heartbeat)#
   ```

   From here, you can configure gateway heartbeat parameters for the network interface. To view all gateway heartbeat parameters, enter a **?** at the system prompt.

5. **state**—Enable or disable the gateway heartbeat feature. The default value is **enabled**. The valid values are:

   - enabled | disabled

6. **heartbeat**—Enter the number of seconds between heartbeats for the media interface gateway. Heartbeats are sent at this interval as long as the media interface is viable. The default value is zero (**0**). The valid range is:

   - Minimum—0
   - Maximum—65535

     The value you configure in this field overrides any globally applicable value set in the gateway heartbeat interval parameter in the Oracle USM HA node (redundancy) configuration.

7. **retry-count**—Enter the number of heartbeat retries that you want sent to the media interface gateway before it is considered unreachable. The default value is zero (**0**). The valid range is:

   - Minimum—0
   - Maximum—65535

8. **retry-timeout**—Enter the heartbeat retry time-out value in seconds. The default value is **1**. The valid range is:

   - Minimum—1
   - Maximum—65535

     This parameter sets the amount of time between system ARP requests to establish media interface gateway communication after a media interface gateway failure.

9. **health-score**—Enter the amount to subtract from the system's health score if a media interface gateway heartbeat fails; this parameter defaults to 0. If the value you set in the retry-time-out field is exceeded, this amount will be subtracted from the system's overall health score. The default value is zero (**0**). The valid range is:

   - Minimum—0
   - Maximum—100

Media State Checkpointing

By default, the Oracle USM performs media checkpointing across the HA node for all signaling protocols. You can keep the default port set for redundancy media flows.

H.323 media high availability is supported through a TCP socket keep-alive, which determines whether or not the other end of a TCP/IP network connection is still in fact connected. This type of checkpointing prevents the listening side of a connection from waiting indefinitely when a TCP connection is lost. When there is a switchover in the HA node, the system that has just become active takes over sending TCP keep-alives. Media continues to flow until the session ends or the flow guard timers expire.

This parameter will appear in the ACLI as follows:

```
red-flow-port                   1985
```

Media State Checkpointing Configuration

To configure media state checkpointing across an HA node in the ACLI:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-manager-config** and press Enter.

   ```
   ORACLE(media-manager)# media-manager-config
   ```

4. **red-flow-port**—Enter the port number for checkpointing media flows associated with the HA interface. This is the port where media flow checkpoint message are sent and received.

   Setting this field to **0** disables media state checkpointing. The default value is **1985**. The valid range is:

   - Minimum—0, 1025
   - Maximum—65535

HA Media Interface Keepalive

In an HA node, it is possible for the two systems in the node to lose communication via the management (rear, wancom) interfaces. For example, wancom 1 and wancom 2 might become disconnected, and cause the heartbeat synchronization to fail. This type of failure causes communication errors because both systems try to assume the active role and thereby access resources reserved for the active system.

To avoid these types of conditions, you can enable an option instructing the standby system to take additional time before going to the active state. This check occurs through the system's media interfaces. Using it, the standby can determine whether or not there has been a true active failure.

In cases when the standby determines the active system has not truly failed, it will go out of service because it will have determined it no longer has up-to-date data from its active counterpart. You can restore functionality by re-establishing management (rear) interface

communication between the system in the node, and then re-synchronizes the standby by rebooting it.

When you enable the media interface keepalive, the standby system in the HA node sends ARP requests to determine if the media interfaces' virtual IP address are active. There are two possible outcomes:

- If it receives responses to its ARP requests, the standby takes itself out of service—to prevent a conflict with the active.
- If it does not receive responses to its ARP requests within a timeout value you set, then standby assumes the active role in the HA node.

Impact to Boot-Up Behavior

With the HA media interface keepalive enabled, the Oracle USM might be in the initial state longer than if the feature were disabled because it requires more information about the media (front) interfaces.

HA Media Interface Keepalive Configuration

You turn the HA media interface keepalive on by setting a timeout value for the standby to receive responses to its ARP requests before it assumes the active role in the HA node. Keeping this parameter set to 0, its default, disables the keepalive

To enable the HA media interface keepalive:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **system** and press Enter.

   ```
   ORACLE(configure)# system
   ORACLE(system)#
   ```

3. Type **redundancy** and press Enter.

   ```
   ORACLE(session-router)# redundancy
   ORACLE(redundancy)#
   ```

   If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **media-if-peercheck-time**—Enter the amount of time in milliseconds for the standby system in an HA node to receive responses to its ARP requests via the media interface before it takes over the active role from its counterpart.

   The default is 0, which turns the HA media interface keepalive off. The maximum value is 500 milliseconds.

5. Save and activate your configuration.

RTC Notes

Starting in Release 4.1, the HA configuration is supported for real-time configuration (RTC). However, not all of the HA-related parameters are covered by RTC because of the impact on service it would cause to reconfigure these parameters dynamically.

This section sets out what parameters you should not dynamically reconfigure, or should dynamically reconfigure with care.

HA

Changes to the following ACLI parameters will have the noted consequences when dynamically reconfigured:

- **cfg-max-trans**—Changing this value could cause the activation time to lengthen slightly
- **init-time**, **becoming-standby-time**, and **becoming-active-time**—Changes take place only if the system is not transitioning between these states; otherwise the system waits until the transition is complete to make changes
- **percent-drift** and **advertisement-time**—Changes are communicated between nodes in the HA pair as part of regular health advertisements

In addition, the following parameters are not part of the RTC enhancement, for the reason specified in the right-hand column.

Parameter	Impact
state	Disrupts service
port	Disrupts service; leaves systems in an HA node without a means of communicating with each other
cfg-port	Disrupts service; leaves systems in an HA node without a means of communicating with each other
cfg-max-trans	Disrupts service
cfg-sync-start-time	Disrupts configuration replication
cfg-sync-comp-time	Disrupts configuration replication

Protocol-Specific Parameters and RTC

In addition, you should not change any of the parameters related to HA that are part of protocol or media management configurations that are used for protocol/media checkpointing. These are:

- SIP configuration
  - **red-max-trans**
  - **red-sync-start-time**
  - **red-sync-comp-time**
- Media Manager configuration
  - **red-flow-port**
  - **red-max-trans**
  - **red-sync-start-time**
  - **red-sync-comp-time**

16

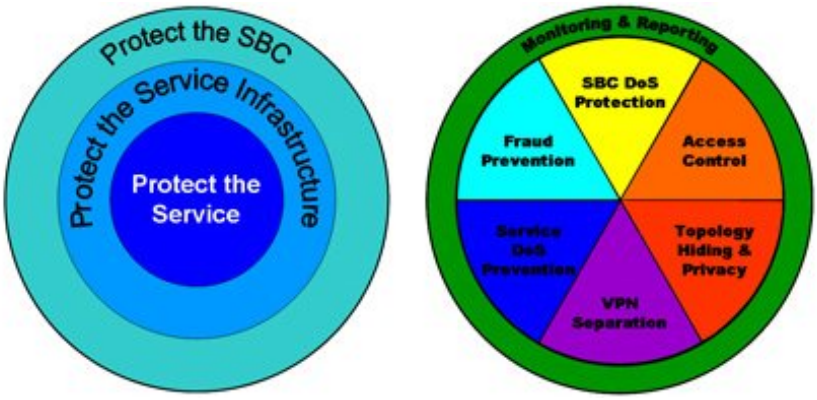
Security

This chapter explains Oracle USM security, which is designed to provide security for administrative security, VoIP and other multimedia services. It includes Admin Security, access control, DoS attack, and overload protection, which help secure service and protect the network infrastructure (including the Oracle USM). In addition, Oracle USM security lets legitimate users still place calls during attack conditions; protecting the service itself.

Security Overview

Oracle USM security includes the Net-SAFE framework's numerous features and architecture designs. Net-SAFE is a requirements framework for the components required to provide protection for the Session Border Controller (SBC), the service provider's infrastructure equipment (proxies, gateways, call agents, application servers, and so on), and the service itself.

The following diagrams illustrate Net-SAFE:



Each of Net-SAFE's seven functions consists of a collection of more specific features:

- Session border controller DoS protection: autonomic, SBC self-protection against malicious and non-malicious DoS attacks and overloads at Layers 2 to 4 (TCP, SYN, ICMP, fragments, and so on) and Layers 5 to 7 (SIP signaling floods, malformed messages, and so on).
- Access control: session-aware access control for signaling and media using static and dynamic permit/deny access control lists (ACLs) at layer 3 and 5.
- Topology hiding and privacy: complete infrastructure topology hiding at all protocol layers for confidentiality and attack prevention security. Also, modification, removal or insertion of call signaling application headers and fields. Includes support for the SIP Privacy RFC.
- VPN separation: support for Virtual Private Networks (VPNs) with full inter-VPN topology hiding and separation, ability to create separate signaling and media-only VPNs, and with optional intra-VPN media hair-pinning to monitor calls within a VPN.
- Service infrastructure DoS prevention: per-device signaling and media overload control, with deep packet inspection and call rate control to prevent DoS attacks from reaching

service infrastructure such as SIP servers, softswitches, application servers, media servers or media gateways.

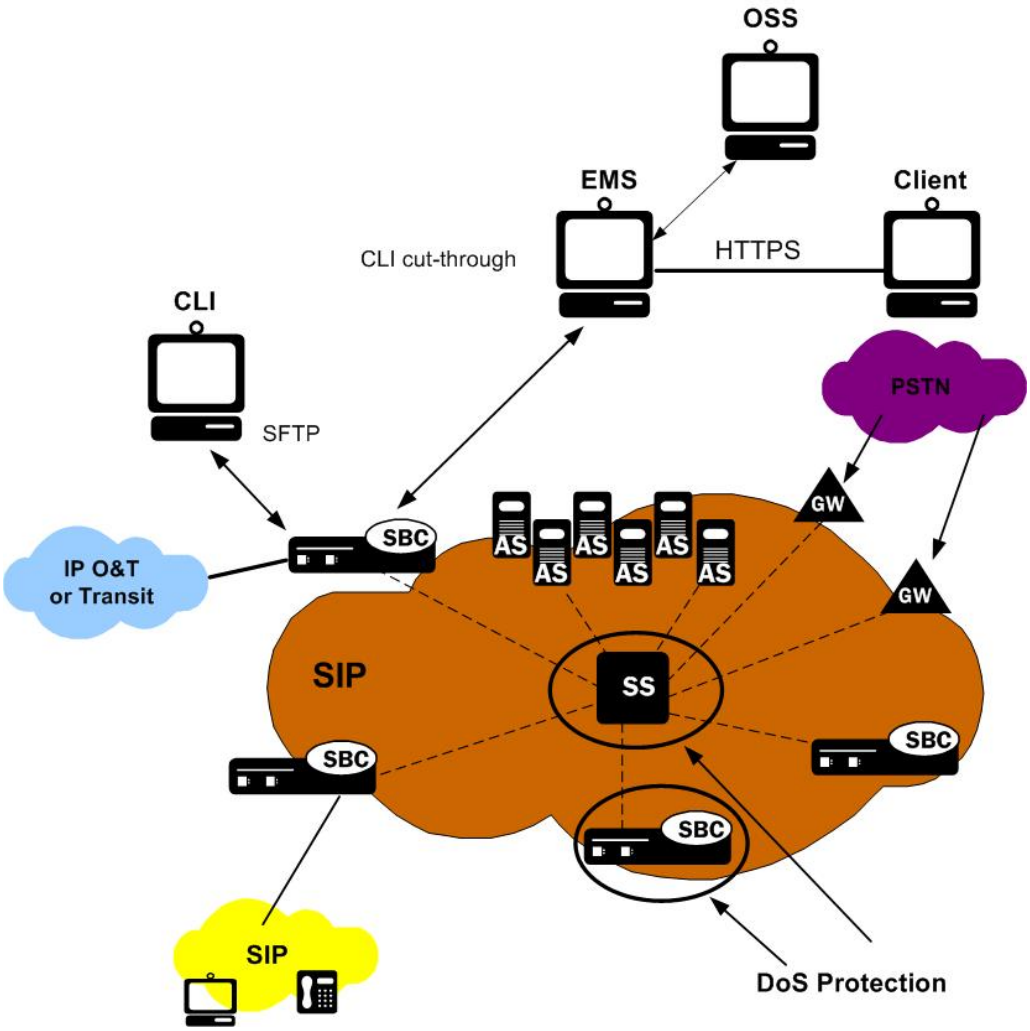
- Fraud prevention: session-based authentication, authorization, and contract enforcement for signaling and media; and service theft protection.
- Monitoring and reporting: audit trails, event logs, access violation logs and traps, management access command recording, Call Detail Records (CDRs) with media performance monitoring, raw packet capture ability and lawful intercept capability. The monitoring method itself is also secured, through the use of SSH and SFTP, and through the ability to use a separate physical Ethernet port for management access.

Denial of Service Protection

This section explains the Denial of Service (DoS) protection for the Oracle USM. The Oracle USM DoS protection functionality protects softswitches and gateways with overload protection, dynamic and static access control, and trusted device classification and separation at Layers 3-5. The Oracle USM itself is protected from signaling and media overload, but more importantly the feature allows legitimate, trusted devices to continue receiving service even during an attack. DoS protection prevents the Oracle USM host processor from being overwhelmed by a targeted DoS attack from the following:

- IP packets from an untrusted source as defined by provisioned or dynamic ACLs
- IP packets for unsupported or disabled protocols
- Nonconforming/malformed (garbage) packets to signaling ports
- Volume-based attack (flood) of valid or invalid call requests, signaling messages, and so on.
- Overload of valid or invalid call requests from legitimate, trusted sources

The following diagram illustrates DoS protection applied to the softswitch and to the Oracle USM.



Levels of DoS Protection

The multi-level Oracle USM DoS protection consists of the following strategies:

- Fast path filtering/access control: access control for signaling packets destined for the Oracle USM host processor as well as media (RTP) packets. The Oracle USM performs media filtering by using the existing dynamic pinhole firewall capabilities. Fast path filtering packets destined for the host processor require the configuration and management of a trusted list and a deny list for each Oracle USM realm (although the actual devices can be dynamically trusted or denied by the Oracle USM based on configuration). You do not have to provision every endpoint/device on the Oracle USM, but instead retain the default values.
- Host path protection: includes flow classification, host path policing and unique signaling flow policing. Fast path filtering alone cannot protect the Oracle USM host processor from being overwhelmed by a malicious attack from a trusted source. The host path and individual signaling flows must be policed to ensure that a volume-based attack will not overwhelm the Oracle USM's normal call processing; and subsequently not overwhelm systems beyond it.
  The Oracle USM must classify each source based on its ability to pass certain criteria that is signaling- and application-dependent. At first each source is considered untrusted with

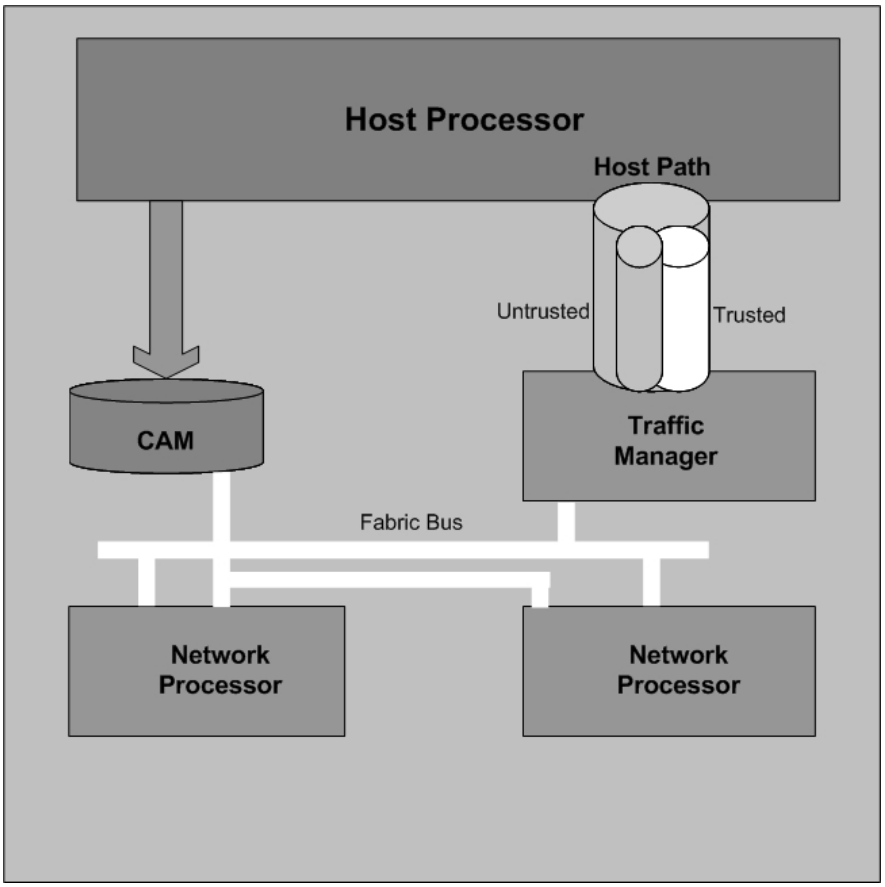
the possibility of being promoted to fully trusted. The Oracle USM maintains two host paths, one for each class of traffic (trusted and untrusted), with different policing characteristics to ensure that fully trusted traffic always gets precedence.

- Host-based malicious source detection and isolation – dynamic deny list. Malicious sources can be automatically detected in real-time and denied in the fast path to block them from reaching the host processor.

About the Process

DoS attacks are handled in the Oracle USM's host path. The Oracle USM uses NAT table entries to filter out undesirable IP addresses; creating a deny list. After a packet from an endpoint is accepted through NAT filtering, policing is implemented in the Traffic Manager subsystem based on the sender's IP address. NAT table entries distinguish signaling packets coming in from different sources for policing purposes. The maximum number of policed calls that the Oracle USM can support is 16K (on 32K CAM / IDT CAM).

The Traffic Manager has two pipes, trusted and untrusted, for the signaling path. Each signaling packet destined for the host CPU traverses one of these two pipes.



Trusted Path

Packets from trusted devices travel through the trusted pipe in their own individual queues. In the Trusted path, each trusted device flow has its own individual queue (or pipe). The Oracle

USM can dynamically add device flows to the trusted list by promoting them from the Untrusted path based on behavior; or they can be statically provisioned.

Trusted traffic is put into its own queue and defined as a device flow based on the following:

- source IP address
- source UDP/TCP port number
- destination IP address
- destination UDP/TCP port (SIP interface to which it is sending)
- realm it belongs to, which inherits the Ethernet interface and VLAN it came in on

For example, SIP packets coming from 10.1.2.3 with UDP port 1234 to the Oracle USM SIP interface address 11.9.8.7 port 5060, on VLAN 3 of Ethernet interface 0:1, are in a separate Trusted queue and policed independently from SIP packets coming from 10.1.2.3 with UDP port 3456 to the same Oracle USM address, port and interface.

Data in this flow is policed according to the configured parameters for the specific device flow, if statically provisioned. Alternatively, the realm to which endpoints belong have a default policing value that every device flow will use. The defaults configured in the realm mean each device flow gets its own queue using the policing values. As shown in the previous example, if both device flows are from the same realm and the realm is configured to have an average rate limit of 10K bytes per second (10KBps), each device flow will have its own 10KBps queue. They are not aggregated into a 10KBps queue.

The individual flow queues and policing lets the Oracle USM provide each trusted device its own share of the signaling, separate the device's traffic from other trusted and untrusted traffic, and police its traffic so that it can't attack or overload the Oracle USM (therefore it is trusted, but not completely).

Address Resolution Protocol Flow

The Address Resolution Protocol (ARP) packets are given their own trusted flow with the bandwidth limitation of 8 Kbps. ARP packets are able to flow smoothly, even when a DoS attack is occurring.

Untrusted Path

Packets (fragmented and unfragmented) that are not part of the trusted or denied list travel through the untrusted pipe. In the untrusted path, traffic from each user/device goes into one of 2048 queues with other untrusted traffic. Packets from a single device flow always use the same queue of the 2048 untrusted queues, and 1/2048th of the untrusted population also uses that same queue. To prevent one untrusted endpoint from using all the pipe's bandwidth, the 2048 flows defined within the path are scheduled in a fair-access method. As soon as the Oracle USM decides the device flow is legitimate, it will promote it to its own trusted queue.

All 2048 untrusted queues have dynamic sizing ability, which allows one untrusted queue to grow in size, as long as other untrusted queues are not being used proportionally as much. This dynamic queue sizing allows one queue to use more than average when it is available. For example, in the case where one device flow represents a PBX or some other larger volume device. If the overall amount of untrusted packets grows too large, the queue sizes rebalance, so that a flood attack or DoS attack does not create excessive delay for other untrusted devices.

In the usual attack situations, the signaling processor detects the attack and dynamically demotes the device to denied in the hardware by adding it to the deny ACL list. Even if the Oracle USM does not detect an attack, the untrusted path gets serviced by the signaling

processor in a fair access mechanism. An attack by an untrusted device will only impact 1/1000th of the overall population of untrusted devices, in the worst case. Even then there's a probability of users in the same 1/1000th percentile getting in and getting promoted to trusted.

IP Fragment Packet Flow

All fragment packets are sent through their own 1024 untrusted flows in the Traffic Manager. The first ten bits (LSB) of the source address are used to determine which fragment-flow the packet belongs to. These 1024 fragment flows share untrusted bandwidth with already existing untrusted-flows. In total, there are 2049 untrusted flows: 1024-non-fragment flows, 1024 fragment flows, and 1 control flow.

Fragmented ICMP packets are qualified as ICMP packets rather than fragment packets. Fragment and non-fragmented ICMP packets follow the trusted-ICMP-flow in the Traffic Manager, with a bandwidth limit of 8Kbs.

Fragment Packet Loss Prevention

You can set the maximum amount of bandwidth (in the **max-untrusted-signaling** parameter) you want to use for untrusted packets. However, because untrusted and fragment packets share the same amount of bandwidth for policing, any flood of untrusted packets can cause the Oracle USM to drop fragment packets.

To prevent fragment packet loss, you can set the **fragment-msg-bandwidth**. When it is set to any value other than 0 (which disables it), the Oracle USM:

- Provides for a separate policing queue for fragment packets (separate from that used for untrusted packets)
- Uses this new queue to prevent fragment packet loss when there is a flood from untrusted endpoints.

When you set up a queue for fragment packets, untrusted packets likewise have their own queue—meaning also that the **max-untrusted-signaling** and **min-untrusted-signaling** values are applied to the untrusted queue.

Static and Dynamic ACL Entry Limits

The Oracle USM can simultaneously police a maximum of 250,000 trusted device flows, while at the same time denying an additional 32,000 attackers. If list space becomes full and additional device flows need to be added, the oldest entries in the list are removed and the new device flows are added.

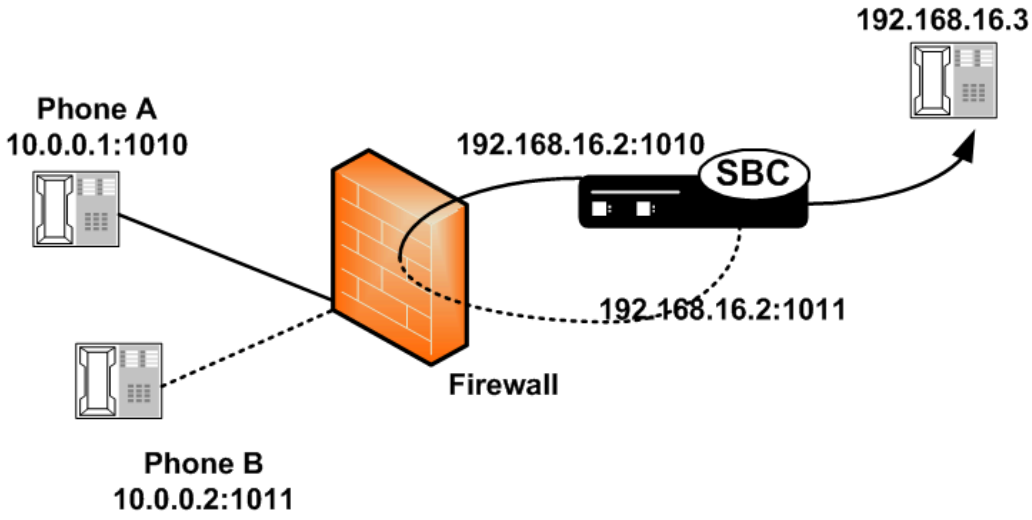
Dynamic Deny for HNT

Dynamic deny for HNT has been implemented on the Oracle USM for cases when callers are behind a NAT or firewall. Without this feature, if one caller behind a NAT or firewall were denied, the Oracle USM would also deny all other users behind the same NAT or firewall. This would be true even for endpoints behind the firewall that had not crossed threshold limits you set for their realm; all endpoints behind the firewall would go out of service. In the following diagram, both Phone A and Phone B would be denied because their IP addresses would be translated by the firewall to the same IPv4 address (192.168.16.2).

However, dynamic deny for HNT allows the Oracle USM to determine, based on the UDP/TCP port, which endpoints should be denied and which should be allowed. The Oracle USM can

determine that even though multiple endpoints originating behind a firewall appear with the same IPv4 address, those addresses use different ports and are unique.

As shown in the diagram below, the ports from Phone A and Phone B remain unchanged. This way, if Phone A violates the thresholds you have configured, the Oracle USM can block traffic from Phone A while still accepting traffic from Phone B.



Host and Media Path Protection Process

The Oracle USM Network Processors (NPs) check the deny and permit lists for received packets, and classify them as trusted, untrusted or denied (discard). Only packets to signaling ports and dynamically signaled media ports are permitted. All other packets sent to Oracle USM ports are filtered. Only packets from trusted and untrusted (unknown) sources are permitted; any packet from a denied source is dropped by the NP hardware. The Traffic Manager manages bandwidth policing for trusted and untrusted traffic, as described earlier. Malicious traffic is detected in the host processor and the offending device is dynamically added to denied list, which enables early discard by the NP. Devices become trusted based on behavior detected by the Signaling Processor, and dynamically added to the trusted list. This process enables the proper classification by the NP hardware. All other traffic is untrusted (unknown).

Session Director Access Control

You an create static trusted/untrusted/deny lists with source IP addresses or IP address prefixes, UDP/TDP port number or ranges, and based on the appropriate signaling protocols. Furthermore, the Oracle USM can dynamically promote and demote device flows based on the behavior, and thus dynamically creates trusted, untrusted, and denied list entries.

Access Control for Hosts

ACLs are supported for all VoIP signaling protocols on the Oracle USM: SIP and H.323. The Oracle USM loads ACLs so they are applied when signaling ports are loaded. The following rules apply to static NAT entries based on your configuration:

- If there are no ACLs applied to a realm that have the same configured trust level as that realm, the Oracle USM adds a default NAT entry using the realm parameters.

- If you configure a realm with none as its trust level and you have configured ACLs, the Oracle USM only applies the ACLs.
- If you set a trust level for the ACL that is lower than the one you set for the realm, the Oracle USM will not add a separate NAT entry for the ACL.

ACLs provide access control based on destination addresses when you configure destination addresses as a way to filter traffic. You can set up a list of access control exceptions based on the source or the destination of the traffic.

For dynamic ACLs based on the promotion and demotion of endpoints, the rules of the matching ACL are applied.

Access Control Endpoint Classification Capacity and DoS

The following capacities are for both IPv4 and IPv6 endpoints.

Platform	Denied	Trusted	Media	Untrusted	Dynamic Trusted	ARP	VLAN
AP4500	32000	8000	32000	2000	250000	4000	4000

Media Access Control

The media access control consists of media path protection and pinholes through the firewall. Only RTP and RTCP packets from ports dynamically negotiated through signaling (SIP and H.323) are allowed, which reduces the chance of RTP hijacking. Media access depends on both the destination and source RTP/RTCP UDP port numbers being correct, for both sides of the call.

Host Path Traffic Management

The host path traffic management consists of the dual host paths discussed earlier:

- Trusted path is for traffic classified by the system as trusted. You can initially define trusted traffic by ACLs, as well as by dynamically promoting it through successful SIP registration, or a successful call establishment. You can configure specific policing parameters per ACL, as well as define default policing values for dynamically-classified flows. Traffic for each trusted device flow is limited from exceeding the configured values in hardware. Even an attack from a trusted, or spoofed trusted, device cannot impact the system.
- Untrusted path is the default for all unknown traffic that has not been statically provisioned otherwise. For example, traffic from unregistered endpoints. Pre-configured bandwidth policing for all hosts in the untrusted path occurs on a per-queue and aggregate basis.

Traffic Promotion

Traffic is promoted from untrusted to trusted list when the following occurs:

- successful SIP registration for SIP endpoints
- successful session establishment for SIP calls

Malicious Source Blocking

Malicious source blocking consists of monitoring the following metrics for each source:

- SIP transaction rate (messages per second)
- SIP call rate (call attempts per second)
- Nonconformance/invalid signaling packet rate

Device flows that exceed the configured invalid signaling threshold, or the configured valid signaling threshold, within the configured time period are demoted, either from trusted to untrusted, or from untrusted to denied classification.

Blocking Actions

Blocking actions include the following:

- Dynamic deny entry added, which can be viewed through the ACLI.
- SNMP trap generated, identifying the malicious source

Dynamically added deny entries expire and are promoted back to untrusted after a configured default deny period time. You can also manually clear a dynamically added entry from the denied list using the ACLI.

Protecting Against Session Agent Overloads

You can prevent session agent overloads with registrations by specifying the registrations per second that can be sent to a session agent.

ARP Flood Protection Enhancements

Enhancements have been made to the way the Oracle USM provides ARP flood protection. In releases prior to Release C5.0, there is one queue for both ARP requests and responses, which the Oracle USM polices at a non-configurable limit (eight kilobytes per second). This method of ARP protection can cause problems during an ARP flood, however. For instance, gateway heartbeats the Oracle USM uses to verify (via ARP) reachability for default and secondary gateways could be throttled; the Oracle USM would then deem the router or the path to it unreachable, decrement the system's health score accordingly. Another example is when local routers send ARP requests for the Oracle USM's address are throttled in the queue; the Oracle USM never receives the request and so never responds, risking service outage.

The solution implemented to resolve this issue is to divide the ARP queue in two, resulting in one ARP queue for requests and a second for responses. This way, the gateway heartbeat is protected because ARP responses can no longer be flooded from beyond the local subnet. In addition, the Oracle USMs in HA nodes generate gateway heartbeats using their shared virtual MAC address for the virtual interface.

In addition, this solution implements a configurable ARP queue policing rate so that you are not committed to the eight kilobytes per second used as the default in prior releases. The previous default is not sufficient for some subnets, and higher settings resolve the issue with local routers sending ARP request to the Oracle USM that never reach it or receive a response.

As a security measure, in order to mitigate the effect of the ARP table reaching its capacity, configuring the media-manager option, **active-arp**, is advised. Enabling this option causes all ARP entries to get refreshed every 20 minutes.

Dynamic Demotion for NAT Devices

In addition to the various ways the Oracle USM already allows you to promote and demote devices to protect itself and other network elements from DoS attacks, it can now block off an entire NAT device. The Oracle USM can detect when a configurable number of devices behind a NAT have been blocked off, and then shut off the entire NAT's access.

This dynamic demotion of NAT devices can be enabled for an access control (ACL) configuration or for a realm configuration. When you enable the feature, the Oracle USM tracks the number of endpoints behind a single NAT that have been labeled untrusted. It shuts off the NAT's access when the number reaches the limit you set.

The demoted NAT device then remains on the untrusted list for the length of the time you set in the **deny-period**.

Configuring DoS Security

This section explains how to configure the Oracle USM for DoS protection.

Configuration Overview

Configuring Oracle USM DoS protection includes masking source IP and port parameters to include more than one match and configuring guaranteed minimum bandwidth for trusted and untrusted signaling path. You can also configure signaling path policing parameters for individual source addresses. Policing parameters are defined as peak data rate (in bytes/sec), average data rate (in bytes/sec), and maximum burst size.

You can configure deny list rules based on the following:

- ingress realm
- source IP address
- source port
- transport protocol (TCP/UDP)
- application protocol (SIP or H.323)

Changing the Default Oracle USM Behavior

The Oracle USM automatically creates permit untrusted ACLs that let all sources (address prefix of 0.0.0.0/0) reach each configured realm's signaling interfaces, regardless of the realm's address prefix. To deny sources or classify them as trusted, you create static or dynamic ACLs, and the global permit untrusted ACL to specifically deny sources or classify them as trusted. Doing this creates a default permit-all policy with specific deny and permit ACLs based on the realm address prefix.

You can change that behavior by configuring static ACLs for realms with the same source prefix as the realm's address prefix; and with the trust level set to the same value as the realm. Doing this prevents the permit untrusted ACLs from being installed. You then have a default deny all ACL policy with specific static permit ACLs to allow packets into the system.

Example 1 Limiting Access to a Specific Address Prefix Range

The following example shows how to install a permit untrusted ACL of source 12.34.0.0/16 for each signalling interface/port of a realm called access. Only packets from within the source address prefix range 12.34.0.0/16, destined for the signaling interfaces/port of the realm named access, are allowed. The packets go into untrusted queues until they are dynamically demoted or promoted based on their behavior. All other packets are denied/dropped.

- Configure a realm called access and set the trust level to low and the address prefix to 12.34.0.0/16.
- Configure a static ACL with a source prefix of 12.34.0.0/16 with the trust level set to low for the realm named access.

Example 2 Classifying the Packets as Trusted

Building on Example 1, this example shows how to classify all packets from 12.34.0.0/16 to the realm signaling interfaces as trusted and place them in a trusted queue. All other packets from outside the prefix range destined to the realm's signaling interfaces are allowed and classified as untrusted; then promoted or demoted based on behavior.

You do this by adding a global permit untrusted ACL (source 0.0.0.0) for each signaling interface/port of the access realm. You configure a static ACL with a source prefix 12.34.0.0/16 and set the trust level to high.

Adding this ACL causes the Oracle USM to also add a permit trusted ACL with a source prefix of 12.34.0.0/16 for each signaling interface/port of the access realm. This ACL is added because the trust level of the ACL you just added is high and the realm's trust level is set to low. The trust levels must match to remove the global permit trusted ACL.

Example 3 Installing Only Static ACLs

This example shows you how to prevent the Oracle USM from installing the global permit (0.0.0.0) untrusted ACL.

- Configure a realm with a trust level of none.
- Configure static ACLs for that realm with the same source address prefix as the realm's address prefix, and set the trust level to any value.

The system installs only the static ACLs you configure.

Access Control List Configuration

To configure access control lists:

1. Access the access-control configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# access-control
ACMEPACKET(access-control)#
```

2. Type select to choose and configure an existing object.

```
ACMEPACKET(access-control)# select
<src-ip>:
1: src 0.0.0.0; 0.0.0.0; realm01; ; ALL
```

3. **realm-id**—Enter the ID of the host's ingress realm.
4. **source-address**—Enter the source IPv4 address and port number for the host in the following format:

   ```
   <IP address>[/number of address bits>][:<port>][/<port bits>]
   ```

   For example:

   ```
   10.0.0.1/24:5000/14
   10.0.0.1/16
   10.0.0.1/24:5000
   10.0.0.1:5000
   ```

   You do not need to specify the number of address bits if you want all 32 bits of the address to be matched. You also do not need to specify the port bits if you want the exact port number matched. If you do not set the port mask value or if you set it to 0, the exact port number will be used for matching. The default value is **0.0.0.0**.
5. **destination-address**—(Is ignored if you configure an application protocol in step 7.) Enter the destination IPv4 address and port for the destination in the following format:

   ```
   <IP address>[/number of address bits>][:<port>[/<port bits>]]
   ```

   You do not need to specify the number of address bits if you want all 32 bits of the address to be matched. You also do not need to specify the port bits if you want the exact port number matched. If you do not set the port mask value or if you set it to 0, the exact port number will be used for matching. The default value is **0.0.0.0**.
6. **application-protocol**—Enter the application protocol type for this ACL entry. The valid values are:
   - SIP | H.323 | None

   > **Note:**
   >
   > If application-protocol is set to none, the destination-address and port will be used. Ensure that your destination-address is set to a non-default value (0.0.0.0.)

7. **transport-protocol**—Select the transport-layer protocol configured for this ACL entry. The default value is **ALL**. The valid values are:
   - ALL | TCP | UDP
8. **access**—Enter the access control type or trusted list based on the trust-level parameter configuration for this host. The default value is **permit**. The valid values are:
   - **permit**—Puts the entry into the untrusted list. The entry is promoted or demoted according to the trust level set for this host.
   - **deny**—Puts the entry in the deny list.
9. **average-rate-limit**—Indicate the sustained rate in bytes per second for host path traffic from a trusted source within the realm. The default value is **0**. A value of **0** means policing is disabled. The valid range is:
   - Minimum—0
   - Maximum—999999999

10. **trust-level**—Indicate the trust level for the host with the realm. The default value is **none**. The valid values are:
    - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
    - **low**—Host can be promoted to the trusted list or demoted to the deny list.
    - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
    - **high**—Host is always trusted.
11. **invalid-signal-threshold**— Enter the number of invalid signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:
    - Minimum—Zero (0) is disabled.
    - Maximum—999999999

      If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

      The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

      The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.
12. **maximum-signal-threshold**—Set the maximum number of signaling messages the host can send within the tolerance window. The value you enter here is only valid when the trust level is low or medium. The default value is **0**, disabling this parameter. The valid range is:
    - Minimum—0
    - Maximum—999999999

      If the number of messages received exceeds this value within the tolerance window, the host is demoted.
13. **untrusted-signal-threshold**—Set the maximum number of untrusted messages the host can send within the tolerance window. Use to configure different values for trusted and untrusted endpoints for valid signaling message parameters. Also configurable per realm. The default value is **0**, disabling this parameter. The valid range is:
    - Minimum—0
    - Maximum—999999999
14. **deny-period**—Indicate the time period in seconds after which the entry for this host is removed from the deny list. The default value is **30**. The valid range is:
    - Minimum—0
    - Maximum—999999999

15. **nat-trust-threshold**—Enter the number of endpoints behind a NAT that must be denied for the Oracle USM to demote the NAT device itself to denied (dynamic demotion of NAT devices). The default is 0, meaning dynamic demotion of NAT devices is disabled. The range is from 0 to 65535.

    The following example shows access control configured for a host in the external realm.

```
access-control
        realm-id                        external
        source-address                  192.168.200.215
        destination-address             192.168.10.2:5000
        application-protocol            SIP
        transport-protocol              ALL
        access                          permit
        average-rate-limit              3343
        trust-level                     low
        invalid-signal-threshold        5454
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     0
```

    The following example of how to configure a black-list entry:

```
access-control
        realm-id                        external
        source-address                  192.168.200.200
        destination-address             192.168.10.2:5000
        application-protocol            SIP
        transport-protocol              ALL
        access                          deny
        average-rate-limit              0
        trust-level                     none
        invalid-signal-threshold        0
        maximum-signal-threshold        0
        untrusted-signal-threshold      0
        deny-period                     0
```

Host Access Policing

You can configure the Oracle USM to police the overall bandwidth of the host path.

To configure host access policing:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

```
ORACLE(configure)# media-manager
```

3. Type **media-manager** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)#
```

4. **max-signaling-bandwidth**—Set the maximum overall bandwidth available for the host path in bytes per second, which includes signaling messages from trusted and untrusted sources. It also includes any Telnet and FTP traffic on media ports. The default value is **1000000**. The valid range is:

    - Minimum—71000

- Maximum—10000000

5. **max-untrusted-signaling**—Set the percentage of the maximum signaling bandwidth you want to make available for messages coming from untrusted sources. This bandwidth is only available when not being used by trusted sources. The default value is **100** The valid range is:
   - Minimum—1
   - Maximum—100
6. **min-untrusted-signaling**—Set the percentage of the maximum signaling bandwidth you want reserved for the untrusted sources. The rest of the bandwidth is available for trusted resources, but can also be used for untrusted sources (see max-untrusted-signaling). The default value is **30** The valid range is:
   - Minimum—1
   - Maximum—100
7. **fragment-msg-bandwidth**—Enter the amount of bandwidth to use for the fragment packet queue. If you leave this parameter set to 0, then the Oracle USM will use the same queue for and share bandwidth between untrusted packets and fragment packets. The default value is zero **(0)**. The valid range is:
   - Minimum—0
   - Maximum—10000000
8. **tolerance-window**—Set the size of the window used to measure host access limits. The value entered here is used to measure the invalid message rate and maximum message rate for the realm configuration. The default value is **30**. The valid range is:
   - Minimum—0
   - Maximum—999999999

   The following example shows a host access policing configuration.

```
media-manager
        state                           enabled
        latching                        enabled
        flow-time-limit                 86400
        initial-guard-timer             300
        subsq-guard-timer               300
        tcp-flow-time-limit             86400
        tcp-initial-guard-timer         300
        tcp-subsq-guard-timer           300
        tcp-number-of-ports-per-flow    2
        hnt-rtcp                        disabled
        algd-log-level                  WARNING
        mbcd-log-level                  WARNING
        home-realm-id
        red-flow-port                   1985
        red-max-trans                   10000
        red-sync-start-time             5000
        red-sync-comp-time              1000
        max-signaling-bandwidth         1000000
        max-untrusted-signaling         50
        min-untrusted-signaling         30
        tolerance-window                30
        rtcp-rate-limit                 0
```

Configuring ARP Flood Protection

You do not need to configure the Oracle USM to enable the use of two separate ARP queues; that feature is enabled automatically.

If you want to configure the ARP queue policing rate, you can do so in the media manager configuration.

> **Note:**
>
> this feature is not RTC-supported, and you must reboot your Oracle USM in order for your configuration changes to take effect.

To set the ARP queue policing rate:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Enter **media-manager** and press <Enter:.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **arp-msg-bandwidth**—Enter the rate at which you want the Oracle USM to police the ARP queue; the value you enter is the bandwidth limitation in bytes per second. The default value is **32000**. The valid range is:

   - Minimum—2000
   - Maximum—200000

5. Save your configuration.
6. Reboot your Oracle USM.

Access Control for a Realm

Each host within a realm can be policed based on average rate, peak rate, and maximum burst size of signaling messages. These parameters take effect only when the host is trusted. You can also set the trust level for the host within the realm. All untrusted hosts share the bandwidth defined for the media manager: maximum untrusted bandwidth and minimum untrusted bandwidth.

To configure access control for a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. **addr-prefix**—Set the IP address prefix used to determine if an IP address is associated with the realm. This value is then associated with the ACLs you create to determine packet access. The default value is **0.0.0.0.**

5. **average-rate-limit**—Set the sustained rate for host path traffic from a trusted source within the realm in bytes per second. The default value is zero (**0**), disabling this parameter. The valid range is:

   - Minimum—0
   - Maximum—4294967295

6. **access-control-trust-level**—Set the trust level for the host within the realm. The default value is **none**. The valid values are:

   - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
   - **low**—Host can be promoted to the trusted list or demoted to the deny list.
   - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
   - **high**—Host is always trusted.

7. **invalid-signal-threshold**— Enter the number of invalid signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:

   - Minimum—Zero (0) is disabled.
   - Maximum—999999999

     If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

     The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

     The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.

8. **maximum-signal-threshold**—Set the maximum number of signaling messages one host can send within the window of tolerance. The host is demoted if the number of messages received by the Oracle USM exceeds the number set here. Valid only when the trust level is set to low or medium. The default value is zero (**0**), disabling this parameter. The valid range is:

   - Minimum—0
   - Maximum—4294967295

9. **untrusted-signal-threshold**—Set the maximum number of untrusted messages the host can send within the tolerance window. Use to configure different values for trusted and untrusted endpoints for valid signaling message parameters. Also configurable per realm. The default value is zero (**0**), disabling the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
10. **deny-period**—Set the length of time an entry is posted on the deny list. The host is deleted from the deny lost after this time period. The default value is **30**. A value of **0** disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295
11. **nat-trust-threshold**—Enter the number of endpoints behind a NAT that must be denied for the Oracle USM to demote the NAT device itself to denied (dynamic demotion of NAT devices). The default is 0, meaning dynamic demotion of NAT devices is disabled. The range is from 0 to 65535.

    The following example shows a host access policing configuration.

```
realm-config
        identifier                      private
        addr-prefix                     192.168.200.0/24
        network-interfaces
                                        prviate:0
        mm-in-realm                     disabled
        mm-in-network                   enabled
        msm-release                     disabled
        qos-enable                      disabled
        max-bandwidth                   0
        ext-policy-svr
        max-latency                     0
        max-jitter                      0
        max-packet-loss                 0
        observ-window-size              0
        parent-realm
        dns-realm
        media-policy
        in-translationid
        out-translationid
        class-profile
        average-rate-limit              8000
        access-control-trust-level      medium
        invalid-signal-threshold        200
        maximum-signal-threshold        0
        untrusted-signal-threshold      500
        deny-period                     30
        symmetric-latching              disabled
        pai-strip                       disabled
        trunk-context
```

Configuring Overload Protection for Session Agents

The Oracle USM offers two methods to control SIP registrations to smooth the registration flow.

You can limit the:

- number of new register requests sent to a session agent (using the **max-register-sustain-rate** parameter)
- burstiness which can be associated with SIP registrations

The first method guards against the Oracle USM's becoming overwhelmed with register requests, while the second method guards against a transient registration that can require more than available registration resources.

SIP registration burst rate control allows you to configure two new parameters per SIP session agent—one that controls the registration burst rate to limit the number of new registration requests, and a second to set the time window for that burst rate. When the registration rate exceeds the burst rate you set, the Oracle USM responds to new registration requests with 503 Service Unavailable messages.

Note that this constraint is not applied to re-registers resulting from a 401 Unauthorized challenge request.

To configure overload protection for session agents:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. **constraints**—Enable this parameter to set the sustained rate window constraint you configure in the next step. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. **sustain-rate-window**—Enter a number to set the sustained window period (in milliseconds) that is used to measure the sustained rate. The default value is zero (**0**). The valid range is:
   - Minimum—10
   - Maximum—4294967295

     The value you set here must be higher than or equal to the value you set for the burst rate window.

     > **Note:**
     >
     > If you are going to use this parameter, you must set it to a minimum value of 10.

6. **max-register-sustain-rate**—Enter a number to set the maximum number of registrations per second you want sent to the session agent. The default value is zero (**0**), disabling the parameter. The valid range is:
   - Minimum—0
   - Maximum—4294967295

7. **register-burst-window**—Define the window size in seconds for the maximum number of allowable SIP registrations. 0 is the minimum and default value for this parameter; the maximum value is 999999999.
8. **max-register-burst-rate**—Enter the maximum number of new registrations you want this session agent to accept within the registration burst rate window. If this threshold is exceeded, the Oracle USM will respond to new registration requests with 503 Service Unavailable messages. 0 is the minimum and default value for this parameter; the maximum value is 999999999.
9. Save and activate your configuration.

DDoS Protection from Devices Behind a NAT

A DDoS attack could be crafted such that multiple devices from behind a single NAT could overwhelm the Oracle USM. The Oracle USM would not detect this as a DDoS attack because each endpoint would have the same source IP but multiple source ports. Because the Oracle USM allocates a different CAM entry for each source IP:Port combination, this attack will not be detected. This feature remedies such a possibility.

Restricting the Number of Endpoints behind a NAT

Each new source IP address and source IP port combination now counts as an endpoint for a particular NAT device. After the configured value of a single NAT's endpoints is reached, subsequent messages from behind that NAT are dropped and the NAT is demoted. This is set with the max-endpoints-per-nat parameter located in both the access-control and realm-config configuration elements.

Counting Invalid Messages from Endpoints behind a NAT

The Oracle USM also counts the number of invalid messages sent from endpoints behind the NAT. Once a threshold is reached, that NAT is demoted. Numerous conditions are counted as Errors/Invalid Messages from an endpoint. The aggregate of all messages from endpoints behind the NAT are counted against the NAT device, in addition to the existing count against the endpoint. This threshold is set with the nat-invalid-message-threshold parameter located in both the access-control and realm-config configuration elements.

As a unique case, the absence of a REGISTER message following a 401 response is counted as an invalid message from the end point. And if that endpoint is behind a NAT, this scenario will be counted as invalid message from that NAT device as well. You set a timeout period in which the REGISTER message must arrive at the Oracle USM. This period is set with the wait-time-for-invalid-register parameter located in the realm config.

DDoS Protection Configuration realm-config

To set the DDoS Protection from devices behind NATs in the realm-config:

1. Access the **realm-config** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Select the **realm-config** object to edit.

```
ORACLE(realm-config)# select
identifier:
1: realm01 left-left:0 0.0.0.0

selection: 1
ORACLE(realm-config)#
```

3. **max-endpoints-per-nat**— Set the maximum number of endpoints that can exist behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
4. **nat-invalid-message-threshold**—Set the maximum number of invalid messages that may originate behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
5. **wait-time-for-invalid-register**—Set the period (in seconds) that the Oracle USM counts before considering the absence of the REGISTER message as an invalid message.
6. Type **done** to save your configuration.

DDoS Protection Configuration access-control

To set the DDoS Protection from devices behind NATs in the access-control configuration element:

1. Access the access-control configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# access-control
ORACLE(access-control)#
```

2. Type select to choose and configure an existing object.

```
ORACLE(access-control)# select
<src-ip>:
1: src 0.0.0.0; 0.0.0.0; realm01; ; ALL
selection:1
```

3. **max-endpoints-per-nat**— Set the maximum number of endpoints that can exist behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
4. **nat-invalid-message-threshold**—Set the maximum number of invalid messages that may originate behind a NAT before demoting the NAT device. Valid values are 0-65535, with 0 disabling this feature. This parameter is also found in the access control configuration element.
5. Type **done** to save your work and continue.

SNMP Trap support

The following trap is sent by the Oracle USM whenever a NAT device is blacklisted due to the triggers listed in this feature. Reasons are not included in the trap, but are available from the syslogs.

```
apSysMgmtExpDOSTrap     NOTIFICATION-TYPE
        OBJECTS         { apSysMgmtDOSIpAddress,  apSysMgmtDOSRealmID ,
                          apSysMgmtDOSFromUri }
        STATUS          deprecated
        DESCRIPTION
```

```
            "This trap is generated when an IP is placed on a deny list due
            to denial-of-service attempts, and provides the ip address that
            has been demoted, the realm-id of that IP, and (if available)
            the URI portion of the SIP From header of the message that
            caused the demotion."
        ::= { apSysMgmtDOSNotifications 2 }
```

Ensure that the enable-snmp-monitor-traps parameter in the system-config configuration element is enabled for the Oracle USM to send out this trap.

Syslog Support

Set the syslog-on-demote-to-deny parameter in the media-manager-config to enabled to generate syslog on endpoint demotion from untrusted to deny. NAT device demotion will also generate a unique syslog message with accompanying text explaining that it is the NAT device demotion event.

Debugging

The show sip acl command now includes counts of Blocked NAT devices.

```
ACMEPACKET# show sipd acl
13:57:28-71
SIP ACL Status            -- Period -- -------- Lifetime --------
                Active    High   Total      Total  PerMax    High
Total Entries        0       0       0          0       0       0
Trusted              0       0       0          0       0       0
Blocked              0       0       0          0       0       0
Blocked NATs         0       0       0          0       0       0
ACL Operations         ---- Lifetime ----
                Recent      Total  PerMax
ACL Requests         0          0       0
Bad Messages         0          0       0
Promotions           0          0       0
Demotions            0          0       0
Trust->Untrust       0          0       0
Untrust->Deny        0          0       0
```

Media Policing

Media policing controls the throughput of individual media flows in the Oracle USM, which in turn provides security and bandwidth management functionality. The media policing feature works for SIP, H.323 and SIP-H.323 protocols. The media policing feature also lets you police static flows and RTCP flows.

The term media policing refers to flows that go through the Oracle USM. Flows that are directed to the host application are not affected by media policing.

You can use media policing to protect against two potential security threats that can be directed against your Oracle USM:

- Media DoS—Once media flows are established through the Oracle USM, network resources are open to RTP media flooding. You can eliminate the threat of a media DoS attack by constraining media flows to absolute bandwidth thresholds.
- Bandwidth Piracy—Bandwidth policing ensures that sessions consume no more bandwidth than what is signaled for.

Policing Methods

The Oracle USM polices real-time traffic by using Constant Bit Rate (CBR) media policing. CBR policing is used when a media flow requires a static amount of bandwidth to be available during its lifetime. CBR policing best supports real-time applications that have tightly constrained delay variation. For example, voice and video streaming are prime candidates for CBR policing.

Session Media Flow Policing

Session media encompasses RTP and RTCP flows. In order to select policing constraints for these flows, the Oracle USM watches for the codec specified in an SDP or H.245 message. When a match is made between the codec listed in an incoming session request and a configured **media-profile** configuration element, the Oracle USM applies that **media-profile**'s bandwidth policing constraint to the media flow about to start.

If multiple codecs are listed in the SDP message, the Oracle USM will use the **media-profile** with the most permissive media policing constraints for all of the flows associated with the session. If a codec in the H.245/SDP message is not found in any configured **media-profile**, the Oracle USM uses the **media-profile** with the most permissive media policing constraints configured. If no **media-profiles** are configured, there will be no session media flow policing.

If a mid-call change occurs, bandwidth policing is renegotiated.

Static Flow Policing

Static flows can also be policed in the same way as media flows are policed. A static flow configuration redirects flows entering the Oracle USM on a media interface. The redirection is based on realm, source, destination, and protocol. When a flow matches the configured static flow criteria, besides being redirected toward a specified destination, its rate can also be controlled based on a static flow policing parameter found in the **static-flow** element. Static flow policing operates obliviously to the data contained within the flow.

Configuration Notes

Review the following information before configuring your Oracle USM to perform media policing.

Session Media Flow Policing

Session media flow policing applies to both RTP and RTCP flows. Setting either of the parameters listed below to 0 disables media policing, letting RTP or RTCP flows pass through the system unrestricted.

- RTP Policing
  - Set in the **media-profile** configuration element's **average-rate-limit** parameter to police RTP traffic with the CBR policing method.
  - **average-rate-limit**—Establishes the maximum speed for a flow in bytes per second.
- RTCP Policing
  - Set in the **media-manager-config** configuration element's **rtcp-rate-limit** parameter to police RTCP traffic with the CBR policing method.

  - **rtcp-rate-limit**—Establishes the maximum speed for an RTCP flow in bytes per second.

Static Flow Policing

Static flow policing is configured with one parameter found in the **static-flow** configuration element. To configure CBR, you have to set the **average-rate-limit** parameter to a non-zero value. Setting the parameter listed below to 0 disables static flow policing, effectively letting the flow pass through the Oracle USM unrestricted.

In a CBR configuration, the **average-rate-limit** parameter determines the maximum bandwidth available to the flow.

- **average-rate-limit**—Establishes the maximum speed for a static flow in bytes per second.

> **Note:**
>
> Static flow policing is not necessarily tied to any type of media traffic, it can affect flows of any traffic type.

Media Policing Configuration for RTP Flows

You can configure media policing in the **media-profile** configuration element using the ACLI. In the following example, you will configure media policing for the G723 media profile.

To configure media policing for RTP flows:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ```

4. Select an existing media profile to which you will add policing constraints.

   ```
   ORACLE(media-profile)# select
   <name>:
   1: audio   4=G723     RTP/AVP 16 0 0 0
   selection:1
   ORACLE(media-profile)#
   ```

   From this point, you can configure media policing parameters. To view all **media-profile** parameters, enter a **?** at the system prompt

5. **average-rate-limit**—Enter the maximum rate in bytes per second for any flows that this **media-profile** polices. The default value is zero (**0**), disabling media policing. The valid range is:

   - Minimum—0
   - Maximum—125000000

     Average rate limit values for common codecs:

- PCMU—80000 Bps
- G729—26000 Bps

The following example shows a **media-profile** configuration element configured for media policing.

```
media-profile
        name                            G723
        media-type                      audio
        payload-type                    4
        transport                       RTP/AVP
        req-bandwidth                   16
        frames-per-packet               0
        parameters
        average-rate-limit              15000
```

Media Policing Configuration for RTCP Flows

You can configure media policing for RTCP flows by using the ACLI.

To configure media policing for RTCP flows:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the **media-manager** path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-manager** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **rtcp-rate-limit**—Enter the RTCP policing constraint in bytes per second. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—125000000

Media Policing Configuration for Static Flows

You can configure media policing for static flows using the ACLI.

To configure media policing for static flows:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the **media-manager** path.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **static-flow** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# static-flow
   ORACLE(static-flow)#
   ```

4. Select an existing static flow to which you will add policing constraints.

```
ORACLE(static-flow)# select
<in-dest-ip>:
1: dest 0.0.0.0; src 192.168.2.1/24; static-flow-in-realm; UDP
selection:1
```

From this point, you can configure media policing parameters for static flows. To view all **static-flow** parameters, enter a **?** at the system prompt

5. **average-rate-limit**—Enter the maximum rate in bytes per second for any flows that this **static-flow** polices. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—125000000

   The following example shows a **static-flow** configuration element configured for media policing.

```
static-flow
        in-realm-id                 static-flow-in-realm
        in-source                   192.168.2.1/24
        in-destination              0.0.0.0
        out-realm-id                static-flow-out-realm
        out-source                  192.168.128.1/24
        out-destination             0.0.0.0
protocol                    UDP
        average-rate-limit          15000
```

RTP Payload Type Mapping

The Oracle USM maintains a default list of RTP payload types mapped to textual encoding names as defined in RFC 3551.

The following table defines the preconfigured payload type for standard encodings.

Payload Type	Encoding Name	Audio (A) / Video (V)	Clock Rate
0	PCMU	A	8000
4	G723	A	8000
8	PCMA	A	8000
9	G722	A	8000
15	G728	A	8000
18	G729	A	8000

If you configure any payload type to encoding name mappings, the default mappings will be ignored. You must then manually enter all payload type mappings you use in the **media-profile** configuration element.

ITU-T to IANA Codec Mapping

The Oracle USM maintains a list of ITU-T (H.245) codecs that map to IANA RTP codecs. An ITU codec is directly mapped to an IANA Encoding Name for media profile lookups. All codecs are normalized to IANA codec names before any matches are made. New ITU-T codecs can not be added to the media profiles list.

The following table defines the ITU-T to IANA codec mappings.

ITU-T	IANA
g711Ulaw64k	PCMU
g711Alaw64k	PCMA
g726	G726
G7231	G723
g728	G728
g729wAnnexB	G729
g729	G729 fmtp:18 annexb=no
H261VideoCapability	H261
H263VideoCapability	H263
t38Fax	T38

SDP Anonymization

In order to provide an added measure of security, the Oracle USM's topology-hiding capabilities include SDP anonymization. Enabling this feature gives the Oracle USM the ability to change or modify certain values in the SDP so that malicious parties will be unable to learn information about your network topology.

To do this, the Oracle USM hides the product-specific information that can appear in SDP o= lines and s= lines. This information can include usernames, session names, and version fields. To resolve this issues, the Oracle USM makes the following changes when you enable SDP anonymization:

- Sets the session name (or the s= line in the SDP) to s=-
- Sets the username in the origin field to -SBC
- Sets the session ID in the origin field to an integer of incrementing value

Note that for mid-call media changes, the session identifier is not incremented.

To enable this feature, you set a parameter in the media manager configuration.

SDP Anonymization Configuration

To enable SDP anonymization:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **media-manager** again to access the media manager configuration, and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **anonymous-sdp**—Set this parameter to enabled to use the SDP anonymization feature. When you leave this parameter empty the feature is turned off. The default value is **disabled**. The valid values are:

   - enabled | disabled

5. Save and activate your configuration.

Unique SDP Session ID

Codec negotiation can be enabled by updating the SDP session ID and version number. The media-manager option, **unique-sdp-id** enables this feature.

With this option enabled, the Oracle USM will hash the session ID and IP address of the incoming SDP with the current date/time of the Oracle USM in order to generate a unique session ID.

Unique SDP Session ID Configuration

To enable unique SDP session ID in media-manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. **options**—Set the options parameter by typing **options**, a Space, the option name **unique-sdp-id** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(media-manager)# options +unique-sdp-id
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the realm configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

4. Save and activate your configuration.

TCP Synchronize Attack Prevention

This section explains how the Oracle USM protects itself from a Transmission Control Protocol (TCP) synchronize (SYN) packet flooding attack sourced from a remote hostile entity.

SIP and H.323 signaling can be configured on the Oracle USM to be TCP protocol-based. In this configuration, the Oracle USM can be a target of a TCP SYN attack. The Oracle USM C is able to service new call requests throughout the duration of an attack

About SYN

SYN is used by TCP when initiating a new connection to synchronize the sequence numbers on two connecting computers. The SYN is acknowledged by a SYN-ACK by the responding computer. After the SYN-ACK, the client finishes establishing the connection by responding with an ACK message. The connection between the client and the server is then open, and the service-specific data can be exchanged between the client and the server.

A SYN flood is a series of SYN packets from forged IP addresses. The IP addresses are chosen randomly and do not provide any hint of the attacker's location. The SYN flood keeps the server's SYN queue full. Normally this would force the server to drop connections. A server that uses SYN cookies, however, will continue operating normally. The biggest effect of the SYN flood is to disable large windows.

Server Vulnerability

Vulnerability to attack occurs when the server has sent a SYN-ACK back to client, but has not yet received the ACK message; which is considered a half-open connection. The server has a data structure describing all pending connections built in its system memory. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections.

The attacking system sends SYN messages to the server that appear to be legitimate, but in fact reference a client that is unable to respond to the SYN-ACK messages. The final ACK message is never sent to the server.

The half-open connections data structure on the server fills and no new incoming connections are accepted until the table is emptied out. Typically there is a timeout associated with a pending connection (the half-open connections will eventually expire and the server will recover). But the attacking system can continue sending IP-spoofed packets requesting new connections faster than the server can expire the pending connections. The server has difficulty in accepting any new incoming network connections.

Configuring TCP SYN Attack Prevention

No configuration is necessary to enable TCP SYN attack prevention. Internal TCP protocol changes were made to provide protection.

Host Certificate Retrieval via SNMP

When a security certificate is installed locally on the Oracle USM , you can poll the expiration of the certificate using the **apSecurityCertificateTable**.

You can configure the Oracle USMto generate the apSecurityCertExpiredNotification trap once a certificate has expired. The number of minutes between notifications sent is configured in the security-config parameter **local-cert-trap-int.**

To send a warning of expiration, you can set the security-config parameter **local-cert-exp-warn-period** to the number of days before the locally installed certificate expires in which you would like a warning.

Host Certificate Retrieval Configuration

To configure the Oracle USM to generate traps when a certificate has or is about to expire:

1. Navigate to the security-config configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security#
   ORACLE(security)# security-config
   ORACLE(security-config)#
   ```

2. Set the **local-cert-exp-warn-period** parameter to the number of days before the locally installed certificate expires in order to receive a warning. A value of 0 disables the trap.

   ```
   ORACLE(security-config)# local-cert-exp-warn-period 3
   ORACLE(security-config)#
   ```

3. Set the **local-cert-trap-int** parameter for the number of minutes between notifications sent once a certificate has expired. A value of 0 disables the warning trap.

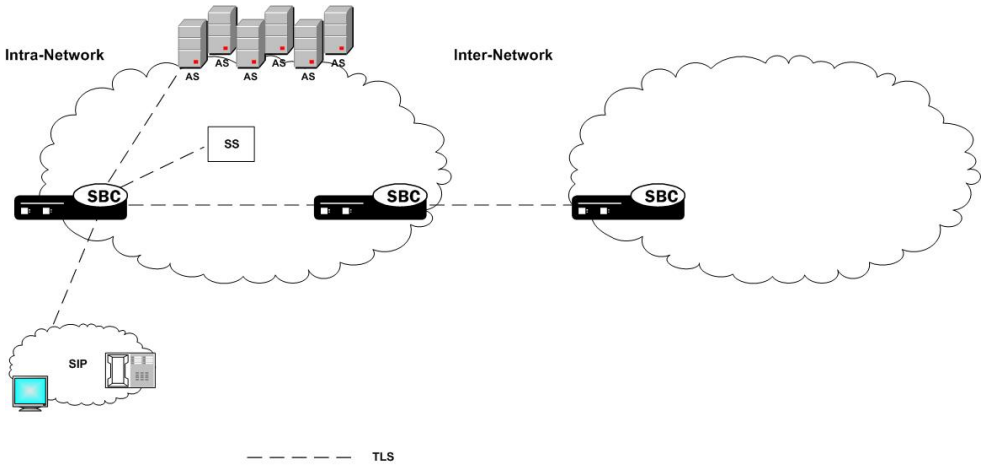
```
ORACLE(security-config)# local-cert-exp-trap-int 15
ORACLE(security-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete required configuration.

Transport Layer Security

The Oracle USM provides support for Transport Layer Security (TLS) for SIP, which can be used to protect user and network privacy by providing authentication and guaranteeing the integrity for communications between the Oracle USM and the following:

- Another device in your network infrastructure (intra-network)
- Another Oracle USM when you are using a peering application (inter-network) for interior network signaling security
- An endpoint for authentication before allowing SIP messaging to take place



The Oracle USM and TLS

The Oracle USM ’s TLS functionality depends on the presence of the Security Service Module (SSM) for hardware acceleration of encryption and decryption and random media generation. The SSM is a plug-on module that can be added to your Oracle USM chassis given the installation of the necessary bootloader and minimum hardware revision levels.

With the requisite hardware revision levels, the plug-on unit can be added to your Oracle USM in the field by qualified personnel. This provision makes upgrades fast, forgoing the need for you to return your Oracle USM to Oracle manufacturing for hardware upgrade. When your Oracle USM is upgraded with the SSM card that supports TLS, a new CLEI code will be added to your chassis; the code will also appear on the SSM card (also referred to as the plug-on unit) and visible if the system’s chassis cover is opened. New Oracle USM s outfitted with the SSM card will have the code labels already affixed in all required locations.

TLS support will not behave in the manner described here if you do not have the SSM component installed on your Oracle USM , because it is the presence of this hardware that enables the TLS software support.

The accelerator card performs:

- RSA
- Diffie-Hellman

- DES
- 3DES
- 40/128 bit ARCFOUR
- AES256
- Random number generation

Supported Encryption

The Oracle USM provides support for TLSv1.2, TLSv1.1, TLSv1.0, and SSLv3 encryption.

> **Note:**
>
> Oracle does not support RC4 ciphers.

TLS Ciphers

The Oracle USM (USM) supports TLS v1, TLSv1.1, TLSv1.2, and SSLv3 ciphers.

For encryption, the USM supports: AES-128, AES-256, 3DES, DES and ARC4 (40 and 128 bit) algorithms. It also supports:

- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_RSA_WITH_DES_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_DES_CBC_SHA
- TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
- TLS_RSA_WITH_NULL_SHA256
- TLS_RSA_WITH_NULL_SHA
- TLS_RSA_WITH_NULL_MD5

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- ALL [default]
- NONE

Mapping SSL3 to TLSv1 Ciphers

The following table shows the mapping of SSL3 ciphers to TLSv1 ciphers:

SSL3	TLSv1
SSL_RSA_WITH_NULL_MD5	TLS_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA	TLS_RSA_WITH_NULL_SHA
SSL_RSA_WITH_RC4_128_MD5	TLS_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA	TLS_DHE_RSA_WITH_DES_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

Note:

The Oracle USM supports TLS_RSA_WITH_NULL_MD5 and TLS_RSA_WITH_NULL_SHA although neither does any encryption. These ciphers might be used for debugging purposes, and should not be deployed.

Minimum Advertised SSL/TLS Version

The sslmin option is available to set a minimum advertised security level to mitigate using older, more vulnerable versions of SSL. One such problem is the poodle attack(CVE-2014-3566).

Oracle USM uses OpenSSL in its SSL/TLS connections. Due to at least one vulnerability, the Poodle attack (CVE-2014-3566), SSLv3 is deemed insecure. Oracle Global Product Security (GPS) suggests that SSLv3 be disabled by default. Setting the option **sslmin** advertises the minimum version the server supports. The **sslmin** option works in conjunction with the tls-profile's **tls-version** parameter when it is set to **compatibility**. For profiles that negotiate to compatible versions, the **sslmin** option specifies the lowest TLS version allowed."

Note:

Note: The next SSL/TLS version after SSLv3 is TLS1.0.

In **security-config**, the **sslmin** option values can be: sslv3, tls1.0, tls1.1 or tls1.2. This change is platform-independent and applies to all Oracle USM.

Minimum Advertised SSL/TLS Version Configuration

Configuring the option **sslmin** to at least tls1.0 for security purposes, provided no **tls-version** in a **tls-profile** requires SSLv3.

1. Access the **security-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# security-config
   ORACLE(security-config)#
   ```

2. Select the **security-config** object to edit.

   ```
   ORACLE(security-config)#

   ORACLE(security-config)#
   ```

3. **options**— Set the options parameter by typing **options**, a space, a plus sign, the option name **sslmin=** and then one of the valid values. Valid values are:

   - sslv3
   - tls1.0
   - tls1.1
   - tls1.2

   ```
   ORACLE(security-config)#options +sslmin=sslv3
   ```

4. Type **done** to save your configuration.

Signaling Support

The Oracle USM ’s TLS functionality supports SIP and SIPS. In addition, the Oracle USM can accommodate a mixture of TLS and non-TLS sessions within a realm as because a request for TLS is controlled by the endpoint (TLS UA).

DoS Protection

The Oracle USM provides the following forms of DoS protection from:

- Too many simultaneous TLS connections being requested by a single IP address.
  The Oracle USM limits the number of TLS connections from a single IP address; you can set a maximum simultaneous number of TCP/TLS connections a SIP interface will allow from a single IP address.
- Too many simultaneous TLS connections being requested by limiting the maximum number of connections for a SIP interface.
  In other words, the maximum simultaneous TCP/TLS connections a SIP interface will allow in aggregate from all IP addresses served by that signaling interface.
- Endpoints establishing TCP/TLS connections that never send any messages (application layer messages; once the TLS handshake completes).
  This protection is triggered by inactivity, measured by lack of any message from a peer. The value specified for this timer is in seconds.
- Endpoints requesting an initial registration that never send messages thereafter.

> **Note:**
>
> It is expected that whenever an endpoint establishes a TCP/TLS connection, it will keep the connection active by sending additional messages or by using the NAT interval configuration. Whenever a connection is torn down because of inactivity, a log at the level "ERROR" is generated.

- Malformed packets by counting and limiting the maximum number of malformed packets. Whenever the Oracle USM receives an invalid TLS message, it increments the internal invalid signalling threshold counter. When that counter reaches the configured value, the Oracle USM denies the endpoints for the configured deny period. This also requires configuration of tolerance window in media manager.

Endpoint Authentication

The Oracle USM does not operate as a CA. Instead, the Oracle USM ’s TLS implementation assumes that you are using one of the standard CAs for generating certificates:

- Verisign
- Entrust
- Thawte
- free Linux-based CA (for example, openssl)

> **Note:**
>
> Self-signed certificates are available only as an option for MSRP connections

The Oracle USM can generate a certificate request in PKCS10 format and to export it. It can also import CA certificates and a Oracle USM certificate in the PKCS7/X509 PEM format.

The Oracle USM generates the key pair for the certificate request internally. The private key is stored as a part of the configuration in 3DES encrypted form (with an internal generated password) and the public key is returned to the user along with other information as a part of PKCS10 certificate request.

The Oracle USM supports the option of importing CA certificates and marking them as trusted. However, the Oracle USM only authenticates client certificates that are issued by the CAs belonging to its trusted list. If you install only a specific vendor's CA certificate on the Oracle USM , it authenticates that vendor's endpoints. Whether the certificate is an individual device certificate or a site-to-site certificate does not matter because the Oracle USM authenticates the signature/public key of the certificate.

Key Usage Control

You can configure the role of a certificate by setting key usage extensions and extended key usage extensions. Both of these are configured in the certificate record configuration.

Key Usage List

This section defines the values you can use (as a list) in the **key-usage-list** parameter. You can configure the parameter with more than one of the possible values.

Value	Description
digitalSignature (default with keyEncipherment)	Used when the subject public key is used with a digital signature mechanism to support security services other than non-repudiation, certificate signing, or revocation information signing. Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity.
nonRepudiation	Used when the subject public key is used to verify digital signatures that provide a non-repudiation service protecting against the signing entity falsely denying some action, excluding certificate or CRL signing.
keyEncipherment (default with digitalSignature)	Used with the subject public key is used for key transport. (For example, when an RSA key is to be used for key management.)
dataEncipherment	Used with the subject public key is used for enciphering user data other than cryptographic keys.
keyAgreement	Used with the subject public key is used key agreement. (For example, when a Diffie-Hellman key is to be used for a management key.)
encipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for enciphering data while performing key agreement.
decipherOnly	The keyAgreement type must also be set. Used with the subject public key is used only for deciphering data while performing key agreement.

Extended Key Usage List

This section defines the values you can use in the **extended-key-usage-list** parameter.

Value	Description
serverAuth (default)	Used while the certificate is used for TLS server authentication. In Oracle USM access-side deployments, the system typically acts as a TLS server accepting TLS connections. You might use this setting while generating the end-entity-cert.
clientAuth	Used while the certificate is used for TLS client authentication. In Oracle USM core-side deployments, the system typically acts as a TLS client initiating TLS connections. You might use this setting while generating the end-entity-cert.

Configuring TLS

This section explains how to configure your Oracle USM for TLS support.

TLS Configuration Process

Configuring Transport Layer Security (TLS) on the Oracle USM (USM) includes the following steps.

1. Obtain and enable the required licences for the platform that you plan to configure for TLS. See "TLS Hardware and License Requirements."
2. Configure certificates. See "Configure Certificates."
3. Configure and apply the TLS profile. See "Configure a TLS Profile."

Configuring Certificates

Configuring certificates is a three-step process:

1. Create a certificate record configuration on the Oracle USM
2. Generate a certificate request by the Oracle USM and save the configuration
3. Import the certificate record into the Oracle USM and save the configuration

Configuring the Certificate Record

The certificate record configuration represents either the end-entity or the CA certificate on the Oracle USM. If it is used to present an end-entity certificate, a private key should be associated with this certificate record configuration using the ACLI **generate-certificate-request** command. A certificate, provided by a CA in response to a certificate request, can be imported to a certificate record configuration using the ACLI **import-certificate** command.

No private key should be associated with the certificate record configuration if it was issued to hold a CA certificate.

> **Note:**
>
> There is no need to create a certificate record when importing a CA certificate or certificate in pkcs12 format.

The following is sample of the certificate record configuration parameters as seen in the ACLI.

```
certificate-record
name            certificate record name
country         country name
state           state name
locality        locality name
organization    organization name
unit            organization unit
common-name     common name
key-size        key size
alternate-name  alternate name
trusted         certificate-record trusted or not
```

To enter a certificate record using the ACLI configuration menu:

1. Access the **certificate-record** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# certificate record
   ORACLE(certificate-record)#
   ```

2. **name**—Provide the required name of this certificate record object.

3. **country**, **state**, **locality**, **organization**, **unit**, and **common-name**—Use these parameters to identify the certificate's subject.

```
ORACLE(certificate-record)# country us
ORACLE(certificate-record)# state ca
ORACLE(certificate-record)# locality “San Francisco”
ORACLE(certificate-record)# organization “Office of the CTO”
ORACLE(certificate-record)# unit cyzygy.com
ORACLE(certificate-record)# common-name www.cyzygy.org/
emailAddress=cto@cyzygy.org
```

   Based on this ACLI sequence, the subject field of the resulting CA-issued certificate reads as follows.

```
Subject: C=US, ST=California, L=San Francisco, O=Office of the CTO
OU=Cyzygy,
CN=www.cyzygy.org/emailAddress=cto@cyzygy.org
```

4. **key-size**—Enter the size of the key for the certificate. The default value is **1024**. The valid range is:

   - 512 | 1024 (default) | 2048

5. **alternate-name**—Optionally provide one or more alternative names for the certificate holder. The Subject Alternative Name certificate extension is defined in section 4.2.1.6 of RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. This extension allows one or more identities to be bound to the subject of a certificate. As a result, each subject alternative name is considered as an alias for the certificate subject. These aliases may be included in addition to, or in place of the identity in the subject field of the certificate, meaning that a certificate can contain a subject name and one or more aliases, or no subject name and one or more aliases. A certificate cannot lack both a subject name and an alias. As specifically defined in RFC 5280, subject alternative names can take the form of an e-mail address, an IP address (IPv4 or IPv6), a DNS address, a Registered ID (RID), a Distinguished Name (DN), or a URI.

   RFC also 5280 allows future support for other alternative name forms, but only if such forms are specified in an IETF RFC.

   Most alternative names take one of the following three formats: IP address (which requires an **IP: prefix**), DNS (which requires a **DNS: prefix**), or email (which requires an **email: prefix**).

   Example:

```
CMEPACKET(certificate-record)# alternate-name IP:192.168.12.101
ACMEPACKET(certificate-record)#

ACMEPACKET(certificate-record)# alternate-name IP:13::17
ACMEPACKET(certificate-record)#

ACMEPACKET(certificate-record)# alternate-name DNS:sg1a.ba.de
ACMEPACKET(certificate-record)#

ACMEPACKET(certificate-record)# alternate-name email:my@other.address
ACMEPACKET(certificate-record)#

ACMEPACKET(certificate-record)# alternate-name copy,email:my@other.address
ACMEPACKET(certificate-record)#
```

   The email form can include a special copy value that includes any email addresses contained in the certificate subject name in the extension.

For other formats, consult RFC 5280, later RFCs that define future formats, or the OpenSSL documentation available at: http://www.openssl.org/docs/apps/x509v3_config.html

6. **trusted**—Leave this parameters set to **enabled** to make the certificate trusted. Enter **disabled** to make this certificate untrusted. The default value is **enabled.** The valid values are:
   - enabled | disabled
7. **key-usage-list**—Enter the usage extensions you want to use with this certificate record. This parameter can be configured with multiple values, and it defaults to the combination of **digitalSignature** and **keyEncipherment**. For a list of possible values and their descriptions, see the section "Key Usage Control" in the *Oracle Communications Session Border Controller Configuration Guide*.
8. **extended-key-usage-list**—Enter the extended key usage extensions you want to use with this certificate record. The default is **serverAuth**. For a list of possible values and their descriptions, see the section "Key Usage Control" in the *Oracle Communications Session Border Controller Configuration Guide*.
9. Type **done** and press Enter.

   ```
   ORACLE(certificate-record)# done
   ```
10. Type **done** to save your configuration.

You create TLS profiles, using your certificate records, to further define the encryption behavior and create the configuration element that you can then apply to a SIP interface.

Generating a Certificate Request

Using the ACLI **generate-certificate-request** command allows you to generate a private key and a certificate request in PKCS10 PEM format. You take this step once you have configured a certificate record.

The Oracle USM stores the private key that is generated in the certificate record configuration in 3DES encrypted form with in internally generated password. The PKCS10 request is displayed on the screen in PEM (Base64) form.

You use this command for certificate record configurations that hold end-entity certificates. If you have configured the certificate record to hold a CA certificate, then you do not need to generate a certificate request because the CA publishes its certificate in the public domain. You import a CA certificate by using the ACLI **import-certificate** command.

This command sends information to the CA to generate the certificate, but you cannot have Internet connectivity from the Oracle USM to the Internet. You can access the internet through a browser such as Internet Explorer if it is available, or you can save the certificate request to a disk and then submit it to the CA.

To run the applicable command, you must use the value you entered in the name parameter of the certificate record configuration. You run the command from main Superuser mode command line:

```
ORACLE# generate-certificate-request acmepacket
Generating Certificate Signing Request. This can take several minutes...
-----BEGIN CERTIFICATE REQUEST-----
MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
```

```
VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY21lMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
-----END CERTIFICATE REQUEST-----;
WARNING: Configuration changed, run save-config command.
ORACLE# save-config
Save-config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot-activate'
ORACLE# activate-config
Activate-Config received, processing.
waiting 12000 for request to finish
Add LI flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt: 0
Request to 'ACTIVATE-CONFIG' has finished
Activate Complete
ORACLE#
```

Importing a Certificate Using the ACLI

For an end-entity certificate, once a certificate is generated using the ACLI generate-certificate-request command, that request should be submitted to a CA for generation of a certificate in PKCS7 or X509v3 format. When the certificate has been generated, it can be imported into the Oracle USM using the **import-certificate** command.

The syntax is:

```
ORACLE # import-certificate [try-all|pkcs7|x509] [certificate-record file-name]
```

To import a certificate:

1. When you use the **import-certificate** command, you can specify whether you want to use PKCS7 or X509v3 format, or try all. In the command line, you enter the command, the format specification, and the name of the certificate record.

   ```
   ORACLE# import-certificate try-all acme
   ```

   The following will appear:

   ```
   Please enter the certificate in the PEM format.
   Terminate the certificate with ";" to exit.......
   -----BEGIN CERTIFICATE-----
   MIIDHzCCAoigAwIBAgIIAhMCUACEAHEwDQYJKoZIhvcNAQEFBQAwcDELMAkGA1UE
   BhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExETAPBgNVBAcTCFNhbiBKb3NlMQ4w
   DAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lwaXQgVGVzdCBDZXJ0aWZpY2F0ZSBB
   dXRob3JpdHkwHhcNMDUwNDEzMjEzNzQzWhcNMDgwNDEyMjEzNzQzWjBUMQswCQYD
   VQQGEwJVUzELMAkGA1UECBMCTUExEzARBgNVBAcTCkJ1cmxpbmd0b24xFDASBgNV
   ```

```
BAoTC0VuZ2luZWVyaW5nMQ0wCwYDVQQDEwRhY21lMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQCXjIeOyFKAUB3rKkKK/+59LT+rlGuW7Lgc1V6+hfTSr0co+ZsQ
bHFUWAA15qXUUBTLJG13QN5VfG96f7gGAbWayfOS9Uymold3JPCUDoGgb2E7m8iu
vtq7gwjSeKNXAw/y7yWy/c04FmUD2U0pZX0CNIR3Mns5OAxQmq0bNYDhawIDAQAB
o4HdMIHaMBEGA1UdEQQKMAiCBnBrdW1hcjAJBgNVHRMEAjAAMB0GA1UdDgQWBBTG
tpodxa6Kmmn04L3Kg62t8BZJHTCBmgYDVR0jBIGSMIGPgBRrRhcU6pR2JYBUbhNU
2qHjVBShtqF0pHIwcDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWEx
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVzaXBpdDEpMCcGA1UECxMgU2lw
aXQgVGVzdCBDZXJ0aWZpY2F0ZSBBdXRob3JpdHmCAQAwDQYJKoZIhvcNAQEFBQAD
gYEAbEs8nUCi+cA2hC/lM49Sitvh8QmpL81KONApsoC4Em24L+DZwz3uInoWjbjJ
QhefcUfteNYkbuMH7LAK0hnDPvW+St4rQGVK6LJhZj7/yeLXmYWIPUY3Ux4OGVrd
2UgV/B2SOqH9Nf+FQ+mNZOlL7EuF4IxSz9/69LuYlXqKsG4=
-----END CERTIFICATE-----;
Certificate imported successfully....
WARNING: Configuration changed, run "save-config" command.
```

2. Save your configuration.

```
ORACLE# save-config
Save-Config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
```

3. Synchronize and activate your configurations.

```
ORACLE# activate-config
Activate-Config received, processing.
waiting 120000 for request to finish
Add LI Flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt:  0
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
ORACLE#
```

Importing a Certificate Using FTP

You can also put the certificate file in the directory /ramdrv and then executing the **import-certificate** command or by pasting the certificate in the PEM/Base64 format into the ACLI. If you paste the certificate, you might have to copy and paste it a portion at a time rather than pasting in the whole thing at once.

To import the certificate using FTP:

1. FTP the certificate file on to the Oracle USM (directory /ramdrv), let us say the name of the certificate file is cert.pem.

2. Once the certificate is successfully transferred to the Oracle USM , run the **import-certificate** command.

The syntax is:

```
ORACLE# import-certificate [try-all|pkcs7|x509] [certificate-record file-name]
```

Using the command will look like this when you have used FTP.

```
ORACLE# import-certificate try-all acme cert.pem
Certificate imported successfully....
WARNING: Configuration changed, run "save-config" command.
```

1. Save your configuration.

```
ORACLE# save-config
Save-Config received, processing.
waiting 1200 for request to finish
Request to 'SAVE-CONFIG' has Finished,
Save complete
Currently active and saved configurations do not match!
To sync & activate, run 'activate-config' or 'reboot activate'.
```

2. Synchronize and activate your configurations.

```
ORACLE# activate-config
Activate-Config received, processing.
waiting 120000 for request to finish
Add LI Flows
LiSysClientMgr::handleNotifyReq
H323 Active Stack Cnt:  0
Request to 'ACTIVATE-CONFIG' has Finished,
Activate Complete
ORACLE#
```

Configure a TLS Profile

The TLS profile configuration contains the information required to run SIP over TLS.

- Obtain the necessary certificates.
- Confirm that the system displays the Superuser mode.

When the Oracle USM (USM) negotiates with TLS, it starts with the highest TLS version and works its way down until it finds a compatible version and cipher that works for the other side.

1. Access the **tls-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

2. **name**—Enter the name of the TLS profile. Required.
3. **end-entity-certificate**—Enter the name of the entity certification record.
4. **trusted-ca-certificates**—Enter the names of the trusted CA certificate records.
5. **cipher-list**—Use either the default **all**, or enter a list of ciphers that you want to support.
6. **verify-depth**—Specify the maximum depth of the certificate chain to verify. Default: 10. Valid range: 0-10.
7. **mutual-authenticate**—Define whether or not you want the USM to mutually authenticate the client. Valid values: enabled | disabled. Default: disabled.
8. **tls-version**—Enter the TLS version that you want to use with this TLS profile. Valid values are:
   - compatibility (default) — When the Oracle Communications Session Border Controller negotiates on TLS, it starts with the highest TLS version and works its way down until it finds a compatible version and cipher that works for the other side.
   - tlsv1
   - tlsv11
   - tlsv12

- SSLv3

> **Note:**
>
> The **security-config > sslmin** option works in conjunction with the tls-profile's **tls-version** parameter when it is set to **compatibility**. For profiles that negotiate to compatible versions, the **sslmin** option specifies the lowest TLS version allowed.

9. Type **done** to save your configuration.

Applying a TLS Profile

To apply the TLS profile, you need to specify it for the SIP interface with which it will be used. You must take this step from within the SIP interface configuration.

1. Type **session-router** and press Enter to access the **session-router** path.

   ```
   ORACLE(configure)# session-router
   ```

2. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

3. Select the existing SIP interface to which you want to apply the TLS profile. If you do not know the same of the profile, press Enter again after you use the select command to see a list of all SIP interfaces. Type in the number corresponding to the SIP interface you want to select, and press Enter. You will then be modifying that SIP interface.

   ```
   ORACLE(sip-interface)# select
   ```

4. **Type sip-ports and** Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-interface)# sip-ports
   ORACLE(sip-port)#
   ```

5. **transport-protocol**—Change the transport protocol to **TLS**.

   ```
   ORACLE(sip-interface)# transport-protocol tls
   ```

6. **tls-profile**—Enter the name of the TLS profile you want applied. This is the same value you enter for the name parameter in the TLS profile configuration. This profile will be applied when the transport protocol is TLS.

   ```
   ORACLE(sip-interface)# tls-profile acmepacket
   ```

7. Save your updated SIP interface configuration.

Reusing a TLS Connection

TheOracle USM supports TLS connection reuse if and when an alias is included in the Via header by the originator of the TLS connection. When this is the case, the Oracle USM reuses the same connection for any outgoing request from the Oracle USM .

Keeping Pinholes Open at the Endpoint

The Oracle USM provides configurable TCP NAT interval on a per-realm basis. You need to configure a NAT interval for the applicable realm to support either all conforming or all non-conforming endpoints.

- Conforming endpoints use the draft-jennings sipping-outbound-01. It describes how to keep the endpoint keeps the connection alive.

  > **Note:**
  >
  > Currently the endpoint uses REGISTER.

- Non-conforming endpoints have short NAT interval, where the HNT application with the TCP connection for TLS operates as it does for regular TCP. We give the UA a shorter expires time so that it refreshes frequently, implicitly forcing the UA to keep the TVP socket open and reuse it for further requests (in-dialog or out-of-dialog). Regular requests using TLS sent from the Oracle USM to the UA reuse the same TCP connection so that further TLS certificate exchanges are not required.

Viewing Certificates

You can view either a brief version or detailed information about the certificates.

Brief Version

Obtaining the brief version uses this syntax, and will appear like the following example:

```
ORACLE# show security certificates brief acmepacket
certificate-record:acmepacket
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            02:13:02:50:00:84:00:71
        Issuer:
            C=US
            ST=California
            L=San Jose
            O=sipit
            OU=Sipit Test Certificate Authority
        Subject:
            C=US
            ST=MA
            L=Burlington
            O=Engineering
            CN=acme
ORACLE#
```

Detailed Version

Obtaining the detailed version uses this syntax, and will appear like the following example:

```
ORACLE# show security certificates detail acmepacket
certificate-record:acmepacket
```

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            02:13:02:50:00:84:00:71
        Signature Algorithm: sha1WithRSAEncryption
        Issuer:
            C=US
            ST=California
            L=San Jose
            O=sipit
            OU=Sipit Test Certificate Authority
        Validity
            Not Before: Apr 13 21:37:43 2005 GMT
            Not After : Apr 12 21:37:43 2008 GMT
        Subject:
            C=US
            ST=MA
            L=Burlington
            O=Engineering
            CN=acme
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:pkumar
            X509v3 Basic Constraints:
                CA:FALSE
ORACLE#
```

Denial of Service for TLS

This section explains the DoS for TLS feature. With this feature, the Oracle USM can provide protection from TCP/TLS message flood by limiting the number of connections from an end point and by limiting the number of simultaneous TCP/TLS connections to a SIP interface.

The Oracle USM protects against a flood of invalid TLS messages and against end points establishing TCP/TLS connections or doing an initial registration without then sending any messages. The Oracle USM protects against:

- Too many simultaneous TLS connections being requested by a single IP address by limiting the number of TLS connections from a single IP address. There is a maximum simultaneous number of TCP/TLS connections a SIP interface will allow from a single IP address.
- Too many simultaneous TLS connections being requested by limiting the maximum number of connections for a SIP interface. There is a maximum number of simultaneous TCP/TLS connections a SIP interface will allow in aggregate from all IP addresses served by that signaling interface.
- End points establishing TCP/TLS connections without then sending any messages (application layer messages post TLS handshake complete). Triggered by inactivity as measured by lack of any message from this peer.
- End points doing an initial registration without then sending any messages.
  This timer could be used by the administrator to detect errors with the SIP configuration. It is expected that whenever an end point establishes a TCP/TLS connection, the end point will keep the connection active by sending messages with REGISTER or by using the NAT interval configuration. Whenever a connection is torn down because of inactivity, a log at the level ERROR is generated.)

- Malformed packets by counting and limiting the maximum number of malformed packets. Whenever an invalid TLS message is received, the internal counter corresponding to invalid-signal-threshold is incremented. When the invalid signal threshold reaches the configured value, the end point will be denied for the configured deny period. (Also requires configuration of the tolerance window in media manager.)
- The max-incoming-conns parameter is well under the maximum number of TLS connections supported by the system. You can set this parameter to it's maximum value of 20000. If you need more than 20000 TLS connections available on this SIP interface, you must set max-incoming-conns to 0 which allows up to the system maximum number of TLS connections, taken on a first come first served basis, on this SIP interface.

DoS for TLS Configuration

You configure the SIP interface and the realm to support DoS for TLS.

DoS protection for TLS Connections on the SIP Interface Configuration

To configure the DoS protection for TCP/TLS connections on a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   From this point, you can configure SIP interface parameters. To view all sip-interface parameters, enter a **?** at the system prompt.

4. **max-incoming-conns**—Enter the maximum number of simultaneous TCP/TLS connections for this SIP interface. The default value is zero (0) which disables any limit to the number of simultaneous TCP/TLS connections on this SIP interface. The valid range is:
   - Minimum—0
   - Maximum—20000
5. **per-src-ip-max-incoming-conns**—Enter the maximum number of connections allowed from an end point.The default value is zero (**0**). The default disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—20000

   > **Note:**
   >
   > To make this parameter effective, you need to set the realm's access-control-trust-level to low or medium.

6. **inactive-conn-timeout**—Enter the time in seconds you want a connection from an endpoint discontinued. This provides protection from end points doing an initial registration without sending any messages. The default value is zero (**0**). The default disables the parameter. The valid range is:
   - Minimum—0
   - Maximum—999999999
7. Save and activate your configuration.

Configuring the SIP Configuration

To configure the SIP configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

   From this point, you can configure SIP configuration parameters. To view all sip-config parameters, enter a **?** at the system prompt.

4. **inactive-dynamic-conn**—Enter the time in seconds after which the Oracle USM tears down inactive dynamic TCP connections. Inactive is defined as not transporting any traffic. This protects against endpoints establishing TCP/TLS connections and then not sending messages. The default value is 32. The valid range is:
   - Minimum—0
   - Maximum—999999999

   > **Note:**
   >
   > Setting this parameter to 0 disables this parameter.

   Because the Oracle USM first establishes a TCP connection, then the TLS connection it waits twice the value entered here after the initiation of a TLS connection before tearing down the connection.

   After an endpoint establishes a TCP/TLS connection, it is supposed to keep the connection active by sending messages or by using the NAT interval configuration. Whenever a connection is torn down because of inactivity, a log at the level ERROR is generated.

Configuring the Realm

To configure the realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

   From this point, you can configure realm parameters. To view all realm configuration parameters, enter a **?** at the system prompt.

4. **deny-period**—Indicate the time period in seconds after which the entry for this host is removed from the deny list. The default value is **30**. The valid range is:

   - Minimum—0
   - Maximum—4294967295

5. **invalid-signal-threshold**— Enter the number of invalid TLS signaling messages that trigger host demotion. The value you enter here is only valid when the trust level is low or medium. Available values are:

   - Minimum—Zero (0) is disabled.
   - Maximum—999999999

     If the number of invalid messages exceeds this value based on the tolerance window parameter, configured in the media manager, the host is demoted.

     The tolerance window default is 30 seconds. Bear in mind, however, that the system uses the same calculation it uses for specifying "recent" statistics in show commands to determine when the number of signaling messages exceeds this threshold. This calculation specifies a consistent start time for each time period to compensate for the fact that the event time, such as a user running a show command, almost never falls on a time-period's border. This provides more consistent periods of time for measuring event counts.

     The result is that this invalid signal count increments for two tolerance windows, 60 seconds by default, within which the system monitors whether or not to demote the host. The signal count for the current tolerance window is always added to the signal count of the previous tolerance window and compared against your setting.

6. **access-control-trust-level**—Set the trust level for the host within the realm. The default value is **none**. The valid values are:

   - **none**—Host is always untrusted. It is never promoted to the trusted list or demoted to the deny list.
   - **low**—Host can be promoted to the trusted list or demoted to the deny list.
   - **medium**—Host can be promoted to the trusted list but is only demoted to untrusted. It is never added to the deny list.
   - **high**—Host is always trusted.

7. Save and activate your configuration.

TLS Session Caching

Transport Layer Security (TLS) session caching allows the Oracle USM to cache key information for TLS connections, and to set the length of time that the information is cached.

When TLS session caching is not enabled, the Oracle USM and a TLS client perform the handshake portion of the authentication sequence in which they exchange a shared secret and encryption keys are generated. One result of the successful handshake is the creation of a unique session identifier. When an established TLS connection is torn down and the client wants to reinstate it, this entire process is repeated. Because the process is resource-intensive, you can enable TLS session caching to avoid repeating the handshake process for previously authenticated clients to preserve valuable Oracle USM resources.

When TLS session caching is enabled on the Oracle USM, a previously authenticated client can request re-connection using the unique session identifier from the previous session. The Oracle USM checks its cache, finds the session identifier, and reinstates the client. This process reduces the handshake to three messages, which preserves system resources.

If the client offers an invalid session identifier, for example, one that the Oracle USM has never seen or one that has been deleted from its cache, the system does not allow the re-connection. The system negotiates the connection as a new connection.

TLS Session Caching Configuration

TLS session caching is global for all TLS functions on your Oracle USM. A new global TLS configuration (**tls-global**) has been added to the system for this purpose.

To enable global TLS session caching:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **security** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **tls-global** and press Enter.

   ```
   ORACLE(security)# tls-global
   ORACLE(tls-global)#
   ```

4. **session-caching**—Set the state for TLS session caching to **enabled** if you want to turn this feature on. The default value is **disabled**. The valid values are:

   - enabled | disabled

5. **session-cache-timeout**—Enter the time in hours that you want the Oracle USM to cache unique session identifiers so that previously authenticated clients can reconnect. The default value is **12**. A value of **0** disables this parameter. The valid range is:

   - Minimum—0
   - Maximum—24

     If you set this parameter to 0, then cache entries will never age (and not be deleted from the cache unless you use the **clear-cache tls** command to delete all entries from the TLS cache). RFC 2246, *The TLS Protocol Version 1.0*, recommends that you set this parameter at the maximum, 24.

TLS Endpoint Certificate Data Caching

To provide a higher level of security for unified messaging (UM), the Oracle USM allows you configure enforcement profiles to cache data from TLS certificates. During the authentication

process, the system caches the data so it can use that data in subsequent SIP message processing. Thus the Oracle USM can:

- Add custom SIP header populated with information from TLS certificates—When the Oracle USM receives an INVITE from a GW, it can write proprietary headers into the SIP message. It uses the certificate information the GW provided during the TLS authentication process with the Oracle USM to do so.
- Compare the host of the Request-URI with information from TLS certificates—When an INVITE is destined for the unified messaging server, the Oracle USM checks the domain of the Request-URI it has generated prior to HMR application. It does so to verify that the Request-URI matches the domain information the UM server provided during the TLS authentication process with the Oracle USM.

TLS endpoint certificate data caching can only applies to call-creating SIP INVITEs. The Oracle USM looks to the following configurations, in order, to apply an enforcement profile: session agent, realm, and SIP interface associated with the INVITE. As a final step, it checks the SIP profile for enforcement profile association.

Inserting Customized SIP Headers in an Outgoing INVITE

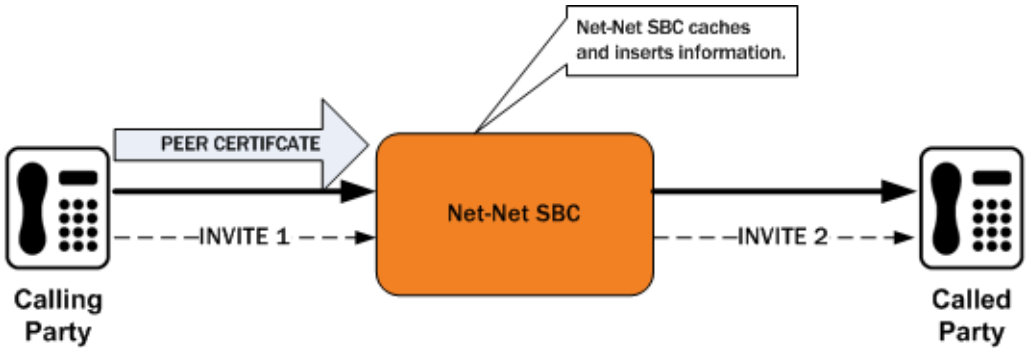
When the Oracle USM establishes a new TLS connection, it caches the following peer certificate attributes:

- Certificate Subject Name
- Certificate Subject Alternative Name (only DNS)

The Oracle USM constructs a customized P-Certificate-Subject-Common-Name SIP header and inserts the header into the outgoing INVITE with the Certificate Subject Name. The Oracle USM also constructs and inserts in the outgoing INVITE one or more P-Certificate-Subject-Alternative-Name SIP headers.

If you enable this capability and the incoming INVITE already has P-Certificate-Subject-Common-Name and P-Certificate-Subject-Alternative-Name headers, the Oracle USM strips them before inserting the new customized ones. It does so to avoid the risk of any attempt to spoof the headers and thereby gain unauthorized access to the UM server.

The following diagram shows a scenario where the calling party establishes a TLS connection with the Oracle USM. Because mutual authentication is enabled, the Oracle USM receives the peer certificate and caches required information from it. This information is inserted in the outgoing INVITE.



The peer certificate from the calling party during the TLS handshake with the Oracle USM looks like the following example.

```
Certificate:
    Data:
```

```
        Version: 3 (0x2)
        Serial Number: 9 (0x9)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=MA, L=Woburn, O=Smith Securities, OU=Certificate
Authority Dept, CN=Smith Certificate Authority/emailAddress=Smith@CA.com
        Validity
            Not Before: Dec 10 21:14:56 2009 GMT
            Not After : Jul 11 21:14:56 2019 GMT
        Subject: C=US, ST=MA, L=Burlington, O=Acme Packet, OU=Certificate
Authority Dept, CN=*.acme.com/emailAddress=ph1Client@acme.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:FALSE
            X509v3 Issuer Alternative Name:
            email:Smith@CA.com
            X509v3 Subject Alternative Name:
            DNS:gw1.acme.com, DNS:gw3.ano.com, DNS:gw2.some.com
            X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
    Signature Algorithm: sha1WithRSAEncryption
```

The outgoing SIP INVITE (INVITE 2 in the diagram) looks like the following sample. Bold text shows where the Oracle USM uses information from the certificate.

```
INVITE sip:222222@acme.com:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.27.113:5060;branch=z9hG4bK4jmg29cmm8l0cg7smmrn85o4q7
From: 111111 <sip:111111@acme.com>;tag=_ph1_tag
To: 222222 <sip:222222@acme.com>
Call-ID: _1-2_call_id-10147@acme.com-1-
CSeq: 1 INVITE
Contact: <sip:111111@172.16.27.113:5060;transport=udp>
P-Certificate-Subject-Common-Name: *.acme.com
P-Certificate-Subject-Alternative-Name: gw1.acme.com
P-Certificate-Subject-Alternative-Name: gw3.ano.com
P-Certificate-Subject-Alternative-Name: gw2.some.com
Max-Forwards: 69
Subject: TBD
Content-Type: application/sdp
Content-Length: 138
Route: <sip:222222@172.16.27.188:5060;lr>
v=0
o=user1 53655765 2353687637 IN IP4 172.16.27.113
s=-
c=IN IP4 172.16.27.113
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Validating the Request-URI Based on Certificate Information

When you configure the Oracle USM to cache TLS certificate information to validate Request-URIs, it stores the Certificate Subject Name and Certificate Subject Alternative Name (only DNS) it learns from peer certificate attributes. It then takes these actions:

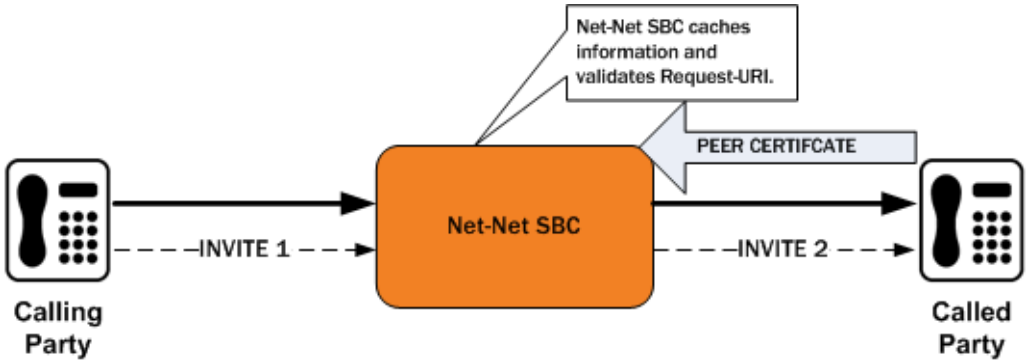
- Extracts the host from the Request-URI of the outgoing INVITE

- Compares this host (exact or wildcard match) with the Certificate Common Name or Certificate Subject Alternative name of the certificate it has received
- Sends out an INVITE if the Certificate Common Name or Certificate Subject Alternative name match; Sends a 403 Forbidden rejection to the endpoint from it received the INVITE if there is no match

Wildcard matching applies only to the prefix of the Request-URI:

```
*.acme.com
    *.*.acmepacket.com
```

This diagram shows a peering scenario where the Oracle USM receives an INVITE from the calling party, which it processes and prepares to send out INVITE 2. After establishing a TLS connection with the called party and caching the required information, the Oracle USM validates the Request-URI. Once validation occurs, the Oracle USM sends INVITE 2.



The peer certificate from the called party during the TLS handshake with the Oracle USM would look like this. Relevant information in the sample appears in **bold** font.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 9 (0x9)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=MA, L=Woburn, O=Smith Securities, OU=Certificate
Authority Dept, CN=Smith Certificate Authority/emailAddress=amith@CA.com
        Validity
            Not Before: Dec 10 21:14:56 2009 GMT
            Not After : Jul 11 21:14:56 2019 GMT
        Subject: C=US, ST=MA, L=Woburn, O=Acme Packet, OU=Certificate Authority
Dept, CN=*.acme.com/emailAddress=ph2Server@acme.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
        X509v3 extensions:
            X509v3 Basic Constraints:
            CA:FALSE
            X509v3 Issuer Alternative Name:
            email:Smith@CA.com
            X509v3 Subject Alternative Name:
            DNS:gw1.acme.com, DNS:*.ano.com, DNS:*.some.com
            X509v3 Key Usage: critical
            Digital Signature, Key Encipherment
    Signature Algorithm: sha1WithRSAEncryption
```

The outgoing SIP INVITE (INVITE 2 in the diagram) would then look like the sample below. The INVITE is sent because **smith.acme.com** matches the common name ***.acme.com**. Other valid SIP Request-URIs would be:

```
222222@gw1.acme.com
222222@smith.ano.com
222222@amith.some.com
```

You can see where the system uses information from the certificate; the text is **bold**.

```
INVITE sip:222222@smith.acme.com:5060 SIP/2.0
Via: SIP/2.0/UDP 172.16.27.113:5060;branch=z9hG4bK4jmg29cmm8l0cg7smmrn85o4q7
From: 111111 <sip:111111@acme.com>;tag=_ph1_tag
To: 222222 <sip:222222@acme.com>
Call-ID: _1-2_call_id-10147@acme.com-1-
CSeq: 1 INVITE
Contact: <sip:111111@172.16.27.113:5060;transport=udp>
Max-Forwards: 69
Subject: TBD
Content-Type: application/sdp
Content-Length: 138
Route: <sip:222222@172.16.27.188:5060;lr>
v=0
o=user1 53655765 2353687637 IN IP4 172.16.27.113
s=-
c=IN IP4 172.16.27.113
t=0 0
m=audio 20000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

TLS Endpoint Certificate Data Caching Configuration

To configure SIP endpoint certificate data caching for an enforcement profile:

1. Access the **enforcement-profile** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# enforcement-profile
   ORACLE(enforcement-profile)#
   ```

2. Select the **enforcement-profile** object to edit.

   ```
   ORACLE(enforcement-profile)# select
   <name>:

   ORACLE(enforcement-profile)#
   ```

3. **add-certificate-info**—Enter a list of one or more certificate attribute names to enable TLS certificate information caching and insertion of cached certificate information into a customized SIP INVITEs. This parameter is empty by default.

   If you want to list more than one value, enclose the value in quotation marks (“ “) and separate the values with Spaces.

   ```
   ORACLE(enforcement-profile)# add-certificate-info "sub-common-name sub-alt-name-DNS"
   ```

4. **certificate-ruri-check**—Change this parameter from **disabled**, its default, to **enabled** if you want your Oracle USM to cache TLS certificate information and use it to validate

Request-URIs. Enabling this parameter also means the Oracle USM will use the cached TLS certificate information in a customized SIP INVITE.

5. Type **done** to save your configuration.

Untrusted Connection Timeout for TCP and TLS

You can configure the Oracle USM for protection against starvation attacks for socket-based transport (TCP or TLS) for SIP access applications. During such an occurrence, the attacker would open a large number of TCP/TLS connections on the Oracle USM and then keep those connections open using SIP messages sent periodically. These SIP messages act as keepalives, and they keep sockets open and consume valuable resources.

Using its ability to promote endpoints to a trusted status, the Oracle USM now closes TCP/TLS connections for endpoints that do not enter the trusted state within the period of time set for the untrusted connection timeout. The attacking client is thus no longer able to keep connections alive by sending invalid messages.

This feature works by setting a value for the connection timeout, which the Oracle USM checks whenever a new SIP service socket for TCP or TLS is requested. If the timer's value is greater than zero, then the Oracle USM starts it. If the timer expires, then the Oracle USM closes the connection. However, if the endpoint is promoted to the trusted state, then the Oracle USM will cancel the timer.

Caveats

This connection timeout is intended for access applications only, where one socket is opened per-endpoint. This means that the timeout is not intended for using in peering applications; if this feature were enabled for peering, a single malicious SIP endpoint might cause the connection to be torn down unpredictably for all calls.

Untrusted Connection Timeout Configuration for TCP and TLS

The untrusted connection timer for TCP and TLS is set per SIP interface.

To set the untrusted connection timer for TCP and TLS:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the signaling-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

   If you are adding support for this feature to a pre-existing SIP configuration, then you must select (using the ACLI **select** command) the configuration that you want to edit.

4. **untrusted-conn-timeout**—Enter the time in seconds that you want the Oracle USM to keep TCP and TLS connections open for untrusted endpoints. The default value is **0**, which will not start the timer. The valid range is:

   - Minimum—0

- Maximum—999999999

5. Save and activate your configuration.

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is defined in RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. The protocol enables users to determine the revocation state of a specific certificate, and may provide a more efficient source of revocation information than is possible with Certificate Revocation Lists (CRL).

The protocol specifies the data exchanged between an OCSP client (for example, the Oracle USM) and an OCSP responder, the Certification Authority (CA), or its delegate, that issued the target certificate. An OCSP client issues a request to an OCSP responder and suspends acceptance of the certificate in question until the responder replies with a certificate status.

Certificate status is reported as

- good
- revoked
- unknown

good indicates a positive response to the status inquiry. At a minimum, this positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval.

revoked indicates that the certificate has been revoked, either permanently or temporarily.

unknown indicates that the responder cannot identify the certificate.

Caveats

OCSP is currently supported only on TLS interfaces; it is not currently supported for use with IKEv1 and IKEv2.

Online Certificate Status Protocol Configuration

OCSP configuration consists of

1. Configuring one or more certificate status profiles; each profile contains information needed to contact a specific OCSP responder.
2. Enabling certificate revocation checking by assigning a certificate status profile to a previously configured TLS profile.

   To create a certificate status profile:

3. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you provide the information required to access one or more OCSP responders.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)#
```

4. Use the required **name** parameter to identify this cert-status-profile instance — each profile instance provides configuration data for a specific OCSP responder. **name** is used to distinguish between multiple profile instances.
5. Use the required **ip-address** parameter to specify the IPv4 address of the OCSP responder.
6. Use the optional **port** parameter to specify the destination port.

   In the absence of an explicitly configured value, the default port number of 80 is used.
7. Use the optional **realm-id** parameter to specify the realm used to transmit OCSP requests.

   In the absence of an explicitly configured value, the default specifies service across the wancom0 interface.
8. Use the optional **requester-cert** parameter only if OCSP requests are signed; ignore this parameter if requests are not signed.

   RFC 2560 does not require signed requests; however, local or CA policies can mandate digital signature..
9. Use the required **responder-cert** parameter to identify the certificate used to validate OCSP responses — a public key of the OCSP responder.

   RFC 2560 requires that all OCSP responders digitally sign OCSP responses, and that OCSP clients validate incoming signatures.

   Provide the name of the certificate configuration element that contains the certificate used to validate the signed OCSP response.
10. Use the optional **retry-count** parameter to specify the maximum number of times to retry an OCSP responder in the event of connection failure.

    If the retry counter specified by this parameter is exceeded, the OCSP requester either contacts another responder (if multiple responders have been configured within this cert-status-profile) and quarantine the unavailable responder for a period defined the **dead-time** parameter.

    In the absence of an explicitly configured value (an integer within the range 0 through 10), the default of 1 is used.

    ```
    ORACLE(cert-status-profile)# retry-count 2
    ORACLE(cert-status-profile)#
    ```
11. Use the optional **dead-time** parameter to specify the quarantine period imposed on an unavailable OCSP responder.

    In the absence of an explicitly configured value (an integer within the range 0 through 3600 seconds), the default value (0) is used.

    Customer implementations utilizing a single OCSP responder are encouraged to retain the default value, or to specify a brief quarantine period to prevent lengthy service outages.
12. Retain default values for the **type** and **trans-protocol** parameter to specify OCSP over an HTTP transport protocol.
13. Use **done**, **exit**, and **verify-config** to complete configuration of this cert-status-profile instance.
14. Repeat Steps 1 through 11 to configure additional certificate status profiles.

    To enable certificate status checking:
15. Move to tls-profile configuration mode.

    ```
    ORACLE# configure terminal
    ORACLE(configure)# security
    ```

```
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

16. Use the required **cert-status-check** parameter to enable OCSP in conjunction with an existing TLS profile.
17. Use the required **cert-status-profile-list** parameter to assign one or more cert-status-profiles to the current TLS profile.

    Each assigned cert-status-profile provides the information needed to access a single OCSP responder.

    Use quotation marks to assign multiple OCSP responders. The following sequence assigns three cert-status-profiles, VerisignClass3Designate, Verisign-1, and Thawte-1 to the TLS-1 profile.
18. Use **done**, **exit**, and **verify-config** to complete configuration.

    Sample Configuration:

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# cert-status-profile
ORACLE(cert-status-profile)# name VerisignClass3Designate
ORACLE(cert-status-profile)# ip-address 192.168.7.100
ORACLE(cert-status-profile)# responder-cert VerisignClass3ValidateOCSP
ORACLE(cert-status-profile)# done
ORACLE(cert-status-profile)# exit
...
...
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)# select
<name>:
1. TLS-1
2. TLS-2
3. TLS-3
selection: 1
ORACLE(tls-profile)# cert-status-check enabled
ORACLE(cert-status-profile)# cert-status-profile-list
VerisignClass3Designate Verisign-1 Thawte-1
ORACLE(cert-status-profile)# done
ORACLE(cert-status-profile)# exit
```

Unreachable OCSR

With OCSP enabled, the client implementation running on the Oracle USM supports message exchange between the Oracle USM and an OCSR as specified in RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. The Oracle USM contacts the OCSR whenever a remote client attempts to establish an SSL/TLS connection with the Oracle USM. The Oracle USM sends a request to the OCSR to check the current status of the certificate presented by the remote client. The Oracle USM suspends processing of the SSL/TLS connection request pending receipt of the OCSR response. In previous releases (prior to Version S-CX6.3.0), a good OCSR response resulted in the establishment of a secure SSL/TLS connection. A revoked or unknown OCSR response, or the failure to reach an OCSR, resulted in the rejection of the connection attempt.

This behavior, which adheres to the requirements of RFC 2560, conflicts with the requirements of Section 5.4.6.2.1.6.4.a.i of UCR 2008 which requires an OCSP client to attempt authentication of remote clients in the event of an unreachable OCSR.

Release S-CX6.3F1 adds a new attribute (**ignore-dead-responder**) to the TLS profile configuration element to provide compliance with DISA/DoD requirements specifying OCSP client operations when faced with unreachable OCSRs. By default, the attribute is disabled meaning that all client connections will be disallowed in the event of unreachable OCSRs.

In DISA/DoD environments **ignore-dead-responder** should be enabled, allowing local certificate-based authentication by the Oracle USM in the event of unreachable OCSRs. Successful authentication is achieved if the certificate presented by the remote client was signed by a Certificate Authority (CA) referenced by the **trusted-ca-certificates** attribute. If the local authentication succeeds, the secure TLS/SSL connection is established; otherwise the connection is rejected.

Unreachable OCSR Configuration

The following sample configuration implements DISA/DoD-compliant client behavior in the event of an unreachable OCSR.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# security#
ACMEPACKET(security)# tls-profile
ACMEPACKET(security)# show
tls-profile
        name                        DoD
        end-entity-certificate      sylarCert-2048
        trusted-ca-certificates     dod1 dod2 disaA disaB IBM1
        cipher-list                 all
        verify-depth                10
        mutual-authenticate         disabled
        tls-version                 tlsv1
        cert-status-check           enabled
        cert-status-profile-list    DoD
        ignore-dead-responder       enabled
        ...
        ...
ACMEPACKET(tls-profile)#
```

OCSR Status Monitoring

OCSR monitoring is provided to track the reachability of individual OCSRs, and, in topologies containing multiple OCSRs, the overall availability of OCSR service.

If monitoring is enabled for individual OCSRs, reachability is monitored by observing responder transactions.

Initially, all OCSRs are considered reachable. If a previously reachable OCSR fails to respond to a certificate status request, the Oracle USM marks the OCSR as unreachable, and generates an SNMP trap and log entry indicating that status. If a previously unreachable OCSR respond to a certificate status request, the Oracle USM returns the OCSR to the reachable status, and generates an SNMP trap and log entry indicating that status change.

Use the following procedure to enable monitoring of individual OCSRs.

1. Navigate to the new security-config configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security#
   ORACLE(security)# security-config
   ORACLE(security-config)#
   ```

2. Enable monitoring of individual OCSRs by setting the **ocsr-monitoring-traps** attribute to **enabled**; this attribute is disabled by default.

```
ORACLE(security-config)# ocsr-monitoring-traps enabled
ORACLE(security-config)#
```

3. Use **done**, **exit**, and **verify-config** to complete required configuration.

   Reachability status of individual OCSRs is aggregated to monitor the overall availability of OCSR service. Using the procedure explained above, the Oracle USM maintains a count of all OCSRs, and of all reachable OCSRs.

   - If all OCSRs are reachable, the Oracle USM generates a trap and log entry noting this optimal state.
   - If all OCSRs are unreachable, the Oracle USM generates a trap and log entry noting this erroneous state.
   - When the Oracle USM transitions from either of the two states described above (in the optimal state, when an OCSR becomes unreachable; in the erroneous state, when an unreachable OCSR becomes reachable), the Oracle USM generates a trap and log entry indicating that an unspecified number of OCSRs are reachable.

   Monitoring of OCSR service availability is a by-product of enabling SNMP; no further configuration is required.

OCSR Access via FQDN

Prior software releases supported OCSR access only via IPv4 addresses and port numbers. In response to a DISA/DoD request, Release S-CX6.3F1 adds support for OCSR access via FQDNs. Since multiple public key infrastructure (PKI) elements capable of supporting OCSP requests can exist within a DISA/DoD environment, the Domain Name Service (DNS) lookup that resolves the FQDN can result in more than one OCSR IP address being returned to the Oracle USM in its role of OCSP client. When processing a lookup that contains more than one IP address, the Oracle USM uses a round-robin algorithm to select from the list of OCSR addresses.

OCSR Access via FQDN is available on all media interfaces and on the wancom0 administrative interface. Note that support for FQDN-based access is requires the configuration of DNS support.

If the **realm** attribute is configured in the certificate-status-profile configuration element, the required DNS query is issued on the corresponding network interface. This model requires configuration of the **dns-ip-primary** attribute, and optionally the **dns-ip-backup1** and **dns-ip-backup2** attributes for the realm's network interface.

If the **realm** attribute is not configured in the certificate-status-profile, the required DNS query is issued on the wancom0 interface. This model requires configuration of the **dns-ip-primary** attribute, and optionally the **dns-ip-backup1** and **dns-ip-backup2** attributes for the wancom0 interface.

Access via an FQDN is supported by a new attribute (**hostname**) in the cert-status-profile configuration element.

The Oracle USM allows configuration of both an OCSR IP address and port number (using the **ip-address** and **port** attributes) and an OCSR domain (using the **hostname** attribute).

In such cases the **verify-config** command issues a warning and notes that IP address-based access will be used.

OCSR Access Configuration via IP Address

The following sample configuration accesses an OCSR at 192.168.7.100:8080.

```
ORACLE# configure terminal
ORACLE(configure)# security#
ORACLE(secirity)# cert-status-profile#
ORACLE(cert-status-profile)# show
cert-status-profile
       name                    defaultOCSP
       ip-address              192.168.7.100
       hostname
       port                    8080
       type                    OCSP
       trans-proto             HTTP
       requestor-cert          ocspVerisignClient
       responder-cert          VerisignCA2
       trusted-cas
       realm-id                admin
       retry-count             1
       dead-time               0
       last-modified-by
       last-modified-date
ORACLE(cert-status-profile)#
```

OCSR Access Configuration via FQDN

The following sample configuration accesses one or more OCSRs at example.disa.mil.

Note that in the absence of a specified domain, the wancom0 interface must be DNS-enabled.

```
ORACLE# configure terminal
ORACLE(configure)# security#
ORACLE(cert-status-profile)# show
cert-status-profile
        name                        DISAdomain2
        ip-address
        hostname                    example.disa.mil
        port
        type                        OCSP
        trans-proto                 HTTP
        requestor-cert
        responder-cert
        trusted-cas                 dod1 dod2 disaA disaB IBM1
        realm-id
        retry-count                 1
        dead-time                   0
        last-modified-by
        last-modified-date
ORACLE(cert-status-profile)#
```

Direct and Delegated Trust Models

RFC 2560 specifies that an OCSR must digitally sign OCSP responses, and that an OCSP client must validate the received signature. In prior releases, successful validation of the signed response served to authenticate the responder. Such an authentication method is referred to as a direct trust model in that it does not require confirmation from a trusted Certificate Authority (CA). Rather it requires that the OCSP client be in possession of the public counterpart of the

private key used by the OCSR to sign the response. This certificate is identified by the **responder-cert** attribute in the cert-status-profile configuration element. Prior to Release S-CX6.3F1, authentication via signature validation was the only authentication method provided by the OCSP client implementation.

Release S-CX6.3F1 continues support for the direct trust model, while also supporting an alternative delegated trust model as described in Section 5.4.6.2.1.6.1.e.3.c of UCR 2010. The delegated trust model requires that OCSR be authenticated by a trusted CA. Within the DISA/DoD delegated trust model, an OCSR certificate is appended to every response, thus eliminating the need for a pre-provisioned responder certificate. The appended certificate is a signing certificate issued and signed by a DoD-approved CA that issued the certificate that is being validated. These OCSR certificates have a short lifespan and are reissued regularly.

Direct Trust Model Configuration

The direct trust model is used in virtually all commercial/enterprise environments. Configuration of the direct trust model is unchanged from that contained in the latest version of your hardware or the Oracle USM*Configuration Guide*.

Delegated Trust Model Configuration

The delegated trust model is used exclusively in some strict DISA/DoD environments; other DISA/DoD environments may support both the direct and delegated trust models.

Use the following procedure to configure OCSP for DISA/DoD environments.

1. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you configure a cert-status-profile configuration element, a container for the information required to access a single, specific OCSR.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# cert-status-profile
   ORACLE(cert-status-profile)#
   ```

2. The **name** attribute differentiates cert-status-profile configuration elements one from another. Each cert-status-profile provides configuration information for a single, specific OCSP responder.
3. The **type** attribute selects the certificate revocation check methodology, the only currently supported methodology is OCSP.
4. Retain the default value (http) for **trans-protocol** attribute, which identifies the transport method used to access the OCSR.
5. The **ip-address** attribute works in conjunction with the **port** attribute to provide the IP address of the OCSR.

   ip-address identifies the OCSR by its IP address. **port** identifies the port monitored by the HTTP server for incoming OCSP requests.

   The **port** attribute can be safely ignored if the OCSR is specified as a FQDN by the **host-name** attribute, but is required if the OCSR is identified by the **ip-address** attribute.

   Allowable **port** values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, the system provides a default value of 80, the well-known HTTP port.
6. Alternatively, use the **host-name** attribute to identify the OCSR.

   host-name identifies the OCSR by a FQDN.

If you provide both an IPv4 address/port number and a FQDN, the Oracle USM uses the IP address/port number and ignores the FQDN.

If values are provided for both attributes, the Security Gateway uses the IP address and ignores the **host-name** value.

7. The **realm-id** attribute specifies the realm used to access the OCSR.

   In the absence of an explicitly configured value, the Oracle USM provides a default value of wancom0, specifying OCSP transmissions across the wancom0 management interface.

   If the OCSR identified by a FQDN, the realm identified by **realm-id** must be DNS-enabled.

8. The **requester-cert** attribute is meaningful only if OCSP requests are signed; ignore this attribute if requests are not signed.

   RFC 2560 does not require the digital signature of OCSP requests. OCSRs, however, can impose signature requirements.

   If a signed request is required by the OCSR, provide the name of the certificate configuration element that contains the certificate used to sign OCSP requests.

9. The **responder-cert** attribute identifies the certificate used to validate signed OCSP response — a public key of the OCSR.

   In DISA/DoD environments that support the direct trust model, optionally provide the name of the certificate configuration element that contains the certificate used to validate the signed OCSP response.

   If a **responder-cert** is provided, it is only used if the OCSP response has no appended certificates, in which case the OCSP client attempts to validate the response signature. Depending on the validation failure or success, the response is rejected or accepted.

   If the OCSP response has an appended certificate or certificate chain, the **responder-cert** is ignored, and the trusted-cas list is used to validate the appended certificate(s).

10. The **trusted-cas** attribute (a list of certificate configuration objects) identifies the approved DoD-approved CAs that sign OCSR certificates.

    In DISA/DoD environments that support the delegated trust model, you must provide a list of CAs used to validate the received certificate.

    If a certificate or a certificate chain is appended to the OCSP response, the OCSP client verifies that the first certificate signed the response, and that the CA is trusted by the Oracle USM (that is, the CA certificate is contained in the **trusted-cas** list. The client then walks through each additional certificate (if any exist) ensuring that each certificate is also trusted. If all certificates are trusted, the OCSP response is accepted; otherwise, it is rejected.

11. The **retry-count** attribute specifies the maximum number of times to retry an OCSP responder in the event of connection failure.

    If the retry counter specified by this attribute is exceeded, the OCSP requester contacts another responder (if multiple responders have been configured) and quarantines the unavailable responder for a period defined the **dead-time** attribute.

    In the absence of an explicitly configured value (an integer within the range 0 through 10), the Oracle USM provides a default value of 1 (connection retries).

12. The **dead-time** attribute specifies the quarantine period imposed on an unavailable OCSR.

    In the absence of an explicitly configured value (an integer within the range 0 through 3600 seconds), the Oracle USM provides a default value of 0 (no quarantine period).

Customer implementations utilizing a single OCSP responder are encouraged to retain the default value, or to specify a brief quarantine period to prevent lengthy service outages.

13. Use **done**, **exit**, and **verify-config** to complete configuration of this cert-status-profile instance.
14. Repeat Steps 1 through 13 to configure additional cert-status-profile configuration elements.

IPv6-IPv4 Internetworking

The Oracle USM supports the following internetworking environments:

SIP-TLS IPv6 endpoints with SIP-TLS IPv4 endpoints

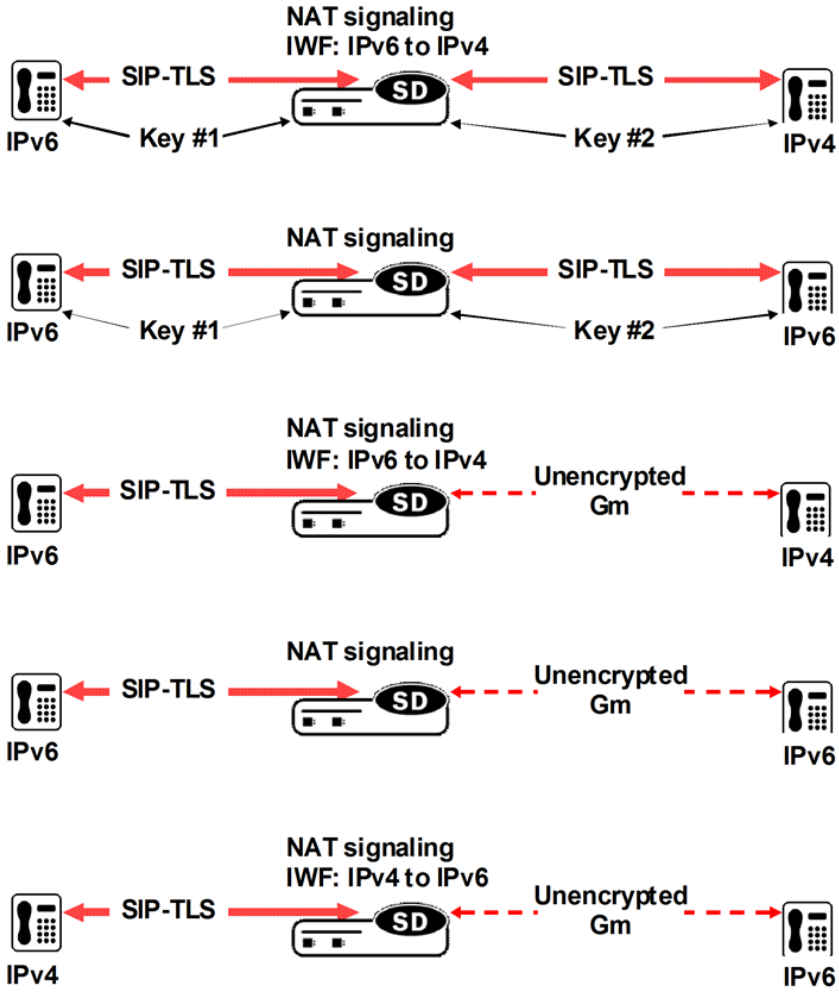
SIP-TLS IPv6 endpoints with SIP-TLS IPv6 endpoints

SIP-TLS IPv6 endpoints with non-SIP-TLS (unencrypted) IPv4 endpoints

SIP-TLS IPv6 endpoints with non-SIP-TLS (unencrypted) IPv6 endpoints

SIP-TLS IPv4 endpoints with non-SIP-TLS (unencrypted) IPv6 endpoints

> **Note:**
>
> Previously delivered TLS functionality, for example, support for certificate extensions, and support for certificate chain processing, is not affected by IPv6-IPv4 internetworking.



TLS SRTP Decryption

The Oracle USM enables the unencrypted replication of incoming and outgoing TLS/SRTP packets and delivery of such unencrypted packets to a CRS.

Decrypted packets are forwarded to the CSR as shown below.

Original encrypted packet:

```
+----------------------+-------------+-----------+------------+
| Encrypted Payload    |    SSL      |   TCP     |   IP       |
|                      |    Header   |   Header  |   Header   |
+----------------------+-------------+-----------+------------+
```

Replicated unencrypted packet:

```
+----------------------+-------------+--------------+-----------+
|   Decrypted Payload  |   UDP       | IP           |  IP       |
|                      |   Header    | Header       |  Header   |
|                      |   (original | (original    |           |
|                      |   src/dest  | src/dest     |           |
|                      |   ports)    | IP addresses |           |
+----------------------+-------------+--------------+-----------+
```

TLS SRTP Decryption Configuration

The **decrypt-tls** attribute enables decryption of TLS/SRTP packets. **decrypt-tls** is disabled by default, meaning that TLS/SRTP forwarded to a CRS are encrypted.

Use the following procedure to enable delivery of unencrypted TLS/SRTP packets to a CRS.

1. Navigate to the new call-recording-server configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router#
ORACLE(session-router)# call-recording-server
ORACLE(call-recording-server)#
```

2. Enable TLS/SRTP decryption by setting the **decrypt-tls** attribute to **enabled**.

```
        ...
        ...
        ...
ORACLE(call-recording-server)# decrypt-tls enabled
ORACLE(call-recording-server)#
        ...
        ...
        ...
ORACLE(call-recording-server)#
```

3. Use **done**, **exit**, and **verify-config** to complete required configuration.

SRTP IPv4 IPv6 Internetworking

Internetworking IPv4 and IPv6 media while using SDES as the key exchange protocol is supported.

SRTP is defined in RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*. It provides confidentiality, message authentication, and replay protection for RTP media and control traffic. SDES is defined in RFC 4568, *Session Description Protocol (SDP) Security Descriptions for Media Streams*. This RFC describes a new SDP cryptographic attribute that provides a secure method to provide security for unicast media streams.

Hardware Requirements

SRTP IPv4 and IPv6 internetworking requires the Enhanced Traffic Controller (ETC) NIU.

Supported Topologies

The following internetworking topologies are supported and illustrated below

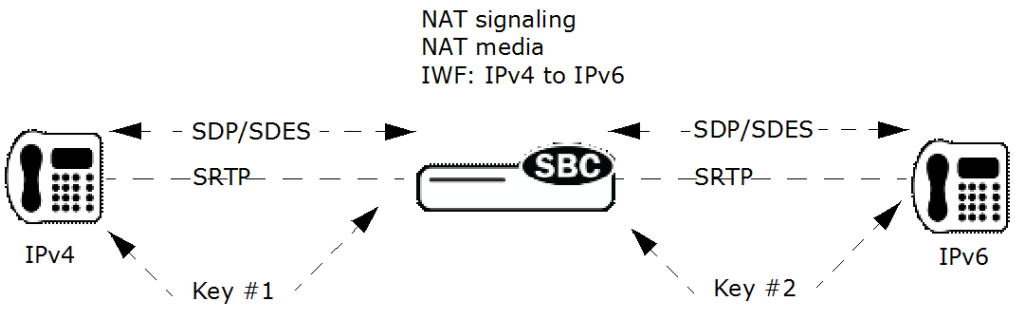
SRTP IPv4 endpoints with SRTP IPv4 endpoints

SRTP IPv4 endpoints with RTP (unencrypted) IPv6 endpoints

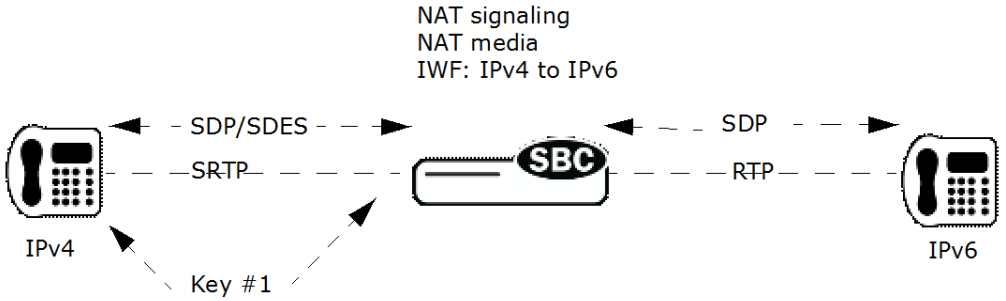
SRTP IPv6 endpoints with SRTP IPv6 endpoints

SRTP IPv6 endpoints with RTP (unencrypted) IPv4 endpoints

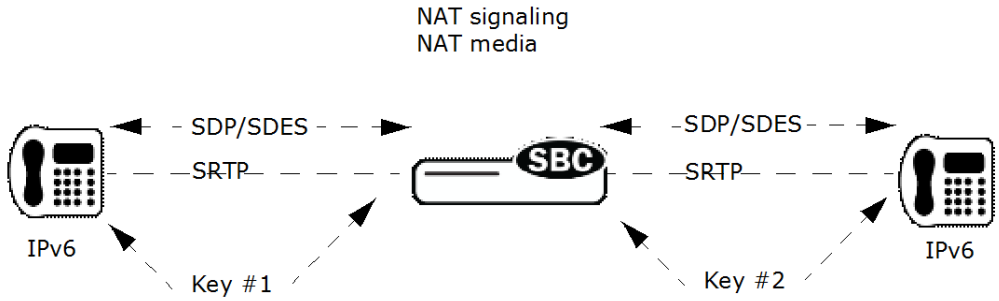
SRTP IPv6 endpoints with RTP (unencrypted) IPv6 endpoints



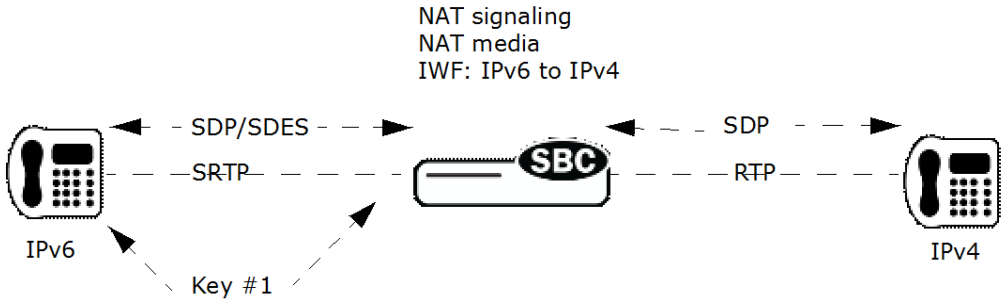
SRTP/SDES IPv4 to SRTP/SDES IPv6 Internetworking



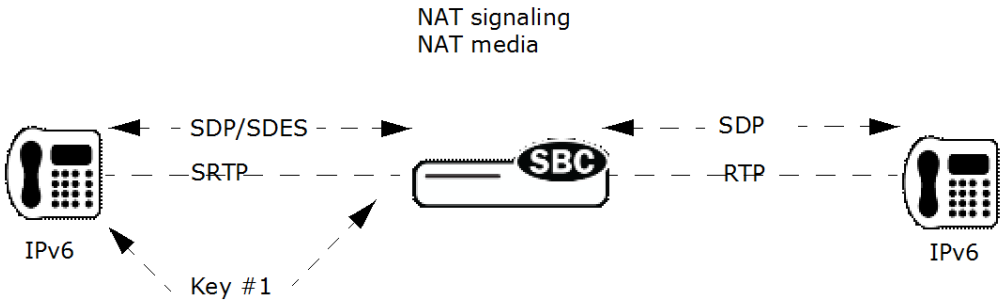
SRTP/SDES IPv4 to RTP IPv6 Internetworking



SRTP/SDES IPv6 to SRTP/SDES IPv6 Internetworking



SRTP/SDES IPv6 to RTP IPv4 Internetworking



SRTP/SDES IPv6 to RTP IPv6 Internetworking

Configuration

IPv6 support must be globally enabled to support SRTP internetworking as described above. If IPv6 is currently enabled, no additional configuration is required.

Key Exchange Protocols

Key exchange protocols enable secure communications over an untrusted network by deriving and distribution shared keys between two or more parties. The Internet Key Exchange (IKEv1) Protocol, originally defined in RFC 2409, provides a method for creating keys used by IPsec tunnels. Session Description Protocol Security Descriptions for Media Streams (SDES), defined in RFC 4568, provides alternative methods for creating keys used to encrypt Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP) transactions.

Each of these protocols is described in the following sections.

IKEv1 Protocol

IKEv1 is specified by a series of RFCs, specifically RFCs 2401 through 2412. The most relevant are:

- RFC 2407, *The Internet IP Security Domain of Interpretation for ISAKMP*
- RFC 2408, *Internet Security Association and Key Management Protocol (ISAKMP)*
- RFC 2409, *The Internet Key Exchange (IKE)*
- RFC 2412, *Oakley Key Determination Protocol*

IKEv1 combines features of the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley Key Determination Protocol in order to negotiate Security Associations (SA) for two communicating peers. IKEv1 also provides for key agreement using Diffie-Hellman.

IKEv1 uses two phases. Phase 1 is used to establish an ISAKMP Security Association for IKEv1 itself. Phase 1 negotiates the authentication method and symmetric encryption algorithm to be used. Phase 1 requires either six messages (main mode) or three messages (aggressive mode).

Phase 2 negotiates the SA for two IPsec peers and is accomplished with three messages.

The initial IKEv1 implementation supports RFC 2409, *Internet Key Exchange*, and RFC 3706, *A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers*.

IKEv1 Configuration

IKEv1 configuration consists of five steps.

1. Configure IKEv1 global parameters.
2. Optionally, enable and configure Dead Peer Detection (DPD) Protocol.
3. Configure IKEv1 interfaces.
4. Configure IKEv1 Security Associations (SA).
5. Assign the IKEv1 SA to an IPsec Security Policy.

IKEv1 Global Configuration

To configure global IKEv1 parameters:

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure global IKEv1 configuration parameters.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ike
   ORACLE(ike)# ike-config
   ORACLE(ike-config)#
   ```

2. Use the **ike-version** parameter to specify IKEv1.

   Use 1 to specify IKEv1 operations.

3. Use the **log-level** parameter to specify the contents of the IKEv1 log.

   Events are listed below in descending order of criticality.

   - emergency (most critical)
   - critical
   - major
   - minor
   - warning
   - notice
   - info (least critical — the default)
   - trace (test/debug, not used in production environments)
   - debug (test/debug, not used in production environments)
   - detail (test/debug, not used in production environments)

   In the absence of an explicitly configured value, the default value of info is used.

4. Use the optional **udp-port** parameter to specify the port monitored for IKEv1 protocol traffic.

   In the absence of an explicitly configured value, the default port number of 500 is used.

5. Use the optional **negotiation-timeout** parameter to specify the maximum interval (in seconds) between Diffie-Hellman message exchanges.

In the absence of an explicitly configured value, the default specifies a 15 second timeout value.

6. Use the optional **event-timeout** parameter to specify the maximum time (in seconds) allowed for the duration of an IKEv1 event, defined as the successful establishment of an IKE or IPsec Security Association (SA).

   In the absence of an explicitly configured value, the default specifies a 60 second time span.

7. Use the optional **phase1-mode** parameter to specify the IKE Phase 1 exchange mode.

   During Phase 1 the IKE initiator and responder establish the IKE SA, using one of two available methods.

   main mode — (the default) is more verbose, but provides greater security in that it does not reveal the identity of the IKE peers. Main mode requires six messages (3 requests and corresponding responses) to (1) negotiate the IKE SA, (2) perform a Diffie-Hellman exchange of cryptographic material, and (3) authenticate the remote peer.

   aggressive mode — is less verbose (requiring only three messages), but less secure in providing no identity protection, and less flexible in IKE SA negotiation.

   In the absence of an explicitly configured value, the default (main mode) is used.

8. Use the optional **phase1-dh-mode** parameter to specify the Diffie-Hellman Group used during IKE Phase 1 negotiation.

   dh-group1 — as initiator, propose Diffie-Hellman group 1 (768-bit primes, less secure)

   dh-group2 — as initiator, propose Diffie-Hellman group 2 (1024-bit primes, more secure)

   first-supported — (the default) as responder, use the first supported Diffie-Hellman group proposed by initiator

9. If functioning as the IKE initiator, use the optional **phase1-life-seconds** parameter to specify the proposed lifetime (in seconds) for the IKE SA established during IKE Phase 1 negotiations.

   Allowable values are within the range 1 through 999999999 (seconds) with a default of 3600 (1 hour).

   This parameter can safely be ignored if functioning as a IKE responder.

10. If functioning as the IKE responder, use the optional **phase1-life-seconds-max** parameter to specify the maximum time (in seconds) accepted for IKE SA lifetime during IKE Phase 1 negotiations.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 86400 (1 day).

    This parameter can safely be ignored if functioning as a IKE initiator.

11. If functioning as the IKE initiator, use the optional **phase2-life-seconds** parameter to specify the proposed lifetime (in seconds) for an IPsec SA established during IKE Phase 2 negotiations.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 28800 (8 hours).

    This parameter can safely be ignored if functioning as a IKE responder.

12. If functioning as the IKE responder, use the optional **phase2-life-seconds-max** parameter to specify the maximum time (in seconds) accepted for IPsec SA lifetime during IKE Phase 2 negotiations.

Allowable values are within the range 1 through 999999999 (seconds) with a default of 86400 (1 day).

This parameter can safely be ignored if functioning as a IKE initiator.

13. Use the optional **phase2-exchange-mode** parameter to specify the Diffie-Hellman group used in Phase 2 negotiations.

    dh-group1 — use Diffie-Hellman group 1 (768-bit primes, less secure)

    dh-group2 — use Diffie-Hellman group 2 (1024-bit primes, more secure)

    no-forward-secrecy — use the same key as used during Phase 1 negotiation

    > **Note:**
    >
    > Forward security indicates that compromise of a single key permits access only to data encrypted with that specific key. Failure to generate a new key for IKE Phase 2 potentially compromises additional data.

    phase1-group — (the default) use the same Diffie-Hellman group as used during Phase 1 negotiation

14. Use the **shared-password** parameter to specify the PSK (pre-shared key) used during authentication with the remote IKE peer.

    The PSK is a string of ACSII printable characters no longer than 255 characters (not displayed by the ACLI).

    This global PSK can be over-ridden by an interface-specific PSK.

15. Use the optional **dpd-time-interval** parameter to specify the maximum period of inactivity before the DPD protocol is initiated on a specific endpoint.

    Allowable values are within the range 1 through 999999999 (seconds) with a default of 0.

    The default value, 0, disables the DPD protocol; setting this parameter to a non-zero value globally enables the protocol and sets the inactivity timer.

16. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 global parameters instance.

DPD Protocol Configuration

If you enabled the DPD protocol with the **dpd-time-interval** parameter, use the following procedure to create a DPD template, an operational set of DPD parameters, that you subsequently assign to one or more IKEv1 interfaces.

To configure DPD parameters:

1. From superuser mode, use the following command sequence to access dpd-params configuration mode. While in this mode, you configure DPD templates.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# ike
ORACLE(ike)# dpd-params
ORACLE(dpd-params)#
```

2. Use the required **name** parameter to provide a unique identifier for this dpd-params instance.

name enables the creation of multiple dpd-params instances.

3. Use the **max-loop** parameter to specify the maximum number DPD peers examined every dpd-interval, whose value is established during IKv1 global configuration.

   If CPU workload surpasses the threshold set by **max-cpu-limit**, this value is over-ridden by **load-max-loop**.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 100.

4. Use the **max-endpoints** parameter to specify the maximum number of simultaneous DPD protocol negotiations supported when the CPU is not under load (as specified by the **max-cpu-limit** property).

   If CPU workload surpasses the threshold set by **max-cpu-limit**, this value is over-ridden by **load-max-endpoints**.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 25.

5. Use the **max-cpu-limit** parameter to specify a threshold value (expressed as a percentage of CPU capacity) at which DPD protocol operations are minimized to conserve CPU resources.

   Allowable values are within the range 0, which effectively disables DPD operations, through 100 (percent) with a default of 60.

6. Use the **load-max-loop** parameter to specify the maximum number of endpoints examined every **dpd-time-interval** when the CPU is under load, as specified by the **max-cpu-limit** parameter.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 40. Ensure that the configured value is less than the value assigned to **max-loop**.

7. Use the **load-max-endpoints** parameter to specify the maximum number of simultaneous DPD Protocol negotiations supported when the CPU is under load, as specified by the **max-cpu-limit** property.

   Allowable values are within the range 1 through 999999999 (endpoints) with a default of 5. Ensure that the configured value is less than the value assigned to **max-endpoints**.

8. Use **done**, **exit**, and **verify-config** to complete configuration of the DPD template instance.

9. Repeat Steps 1 through 8 to configure additional DPD templates.

IKEv1 Interface Configuration

To configure IKEv1 interface parameters:

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure IKEv1 interface parameters.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ike
   ORACLE(ike)# ike-interface
   ORACLE(ike-interface)#
   ```

2. Use the **address** parameter to specify the IPv4 address of the interface.

3. Use the **realm-id** parameter to specify the realm that contains the IP address assigned to this IKEv1 interface.

4. Use the **ike-mode** parameter to specify the operational mode, either responder (the default) or initiator.
5. If DPD has been enabled at the global level, use the **dpd-params-name** parameter to assign a DPD template, an operational set of DPD parameters, to the current IKEv1 interface.

   If DPD has not been enabled, this parameter can be safely ignored.
6. Use the optional **shared-password** parameter to assign an interface PSK.

   This IKEv1-interface-specific value over-rides the global default value set at the IKE configuration level.
7. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 interface.
8. Repeat Steps 1 through 7 to configure additional IKEv1 interfaces.

IKEv1 Security Association Configuration

An IKEv1 SA identifies cryptographic material available for IPsec tunnel establishment.

To configure IKEv1 SA parameters:

1. From superuser mode, use the following command sequence to access ike-sainfo configuration mode. While in this mode, you configure global IKEv1 SAs.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ike
   ORACLE(ike)# ike-sainfo
   ORACLE(ike-sainfo)#
   ```
2. Use the required **name** parameter to provide a unique identifier for this ike-sainfo instance.

   name enables the creation of multiple ike-sainfo instances.
3. Use the **security-protocol** parameter to specify the IPsec security (authentication and encryption) protocols supported by this SA.

   The following security protocols are available.

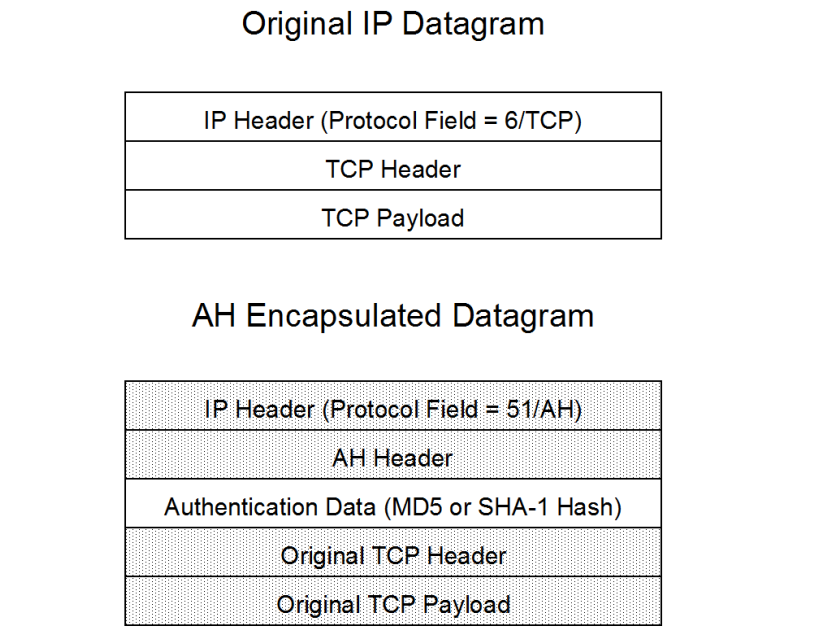
   Authentication Header (AH) — the default value — as defined by RFC 4302, *IP Authentication Header*, which provides authentication integrity to include the mutual identification of remote peers, non-repudiation of received traffic, detection of data that has been altered in transit, and detection of data that has been replayed, that is copied and then re-injected into the data stream at a later time. Authentication services utilize the authentication algorithm specified by the **auth-algo** property.

   Encapsulating Security Payload (ESP) as defined by RFC 4303, *IP Encapsulating Security Payload*, which provides both authentication and privacy services. Privacy services utilize the encryption algorithm specified by the **encryption-algo** property.

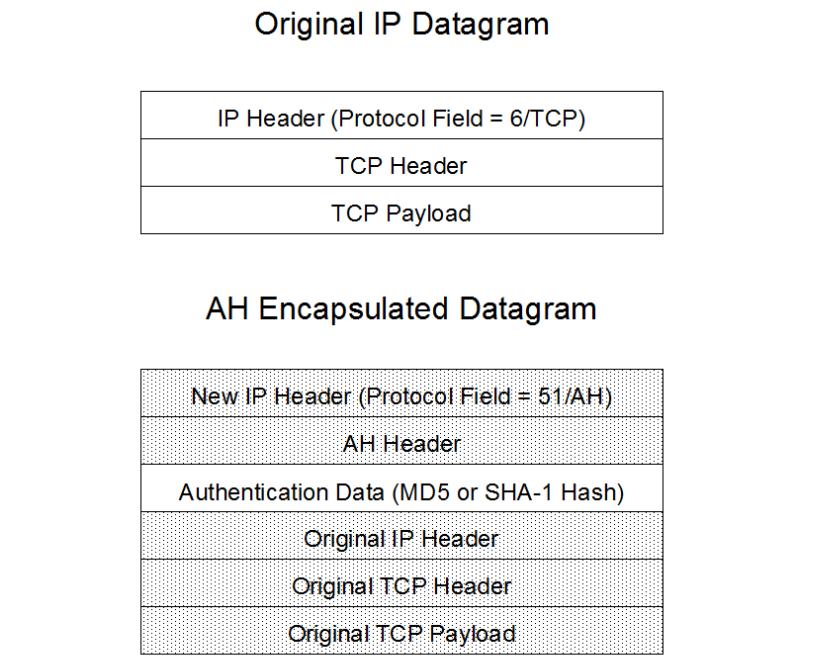
   ESP-AUTH (also RFC 4303-based), which supports ESP's optional authentication.

   ESP-NULL (also RFC 4303-based) which proves NULL encryption as described in RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*. This option provides no privacy services, and is not recommended for production environments.

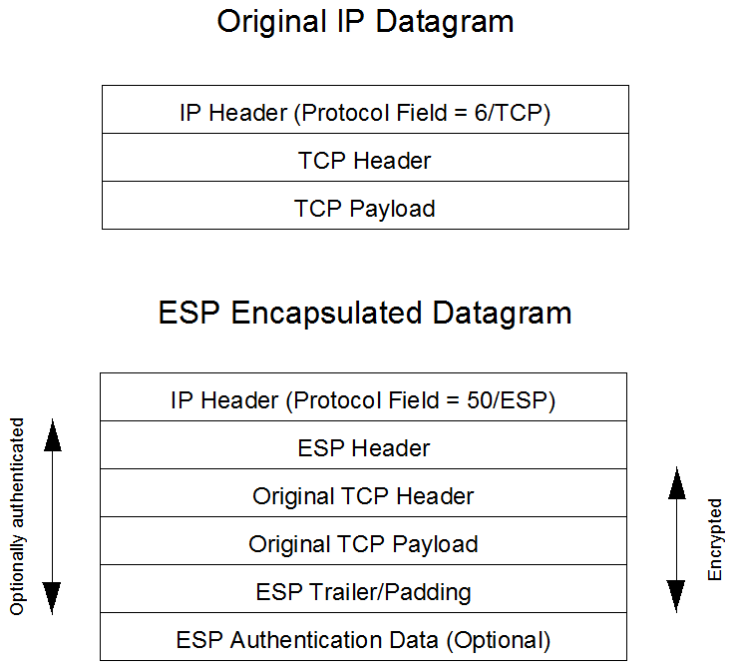
   Refer to the following figures for additional details.



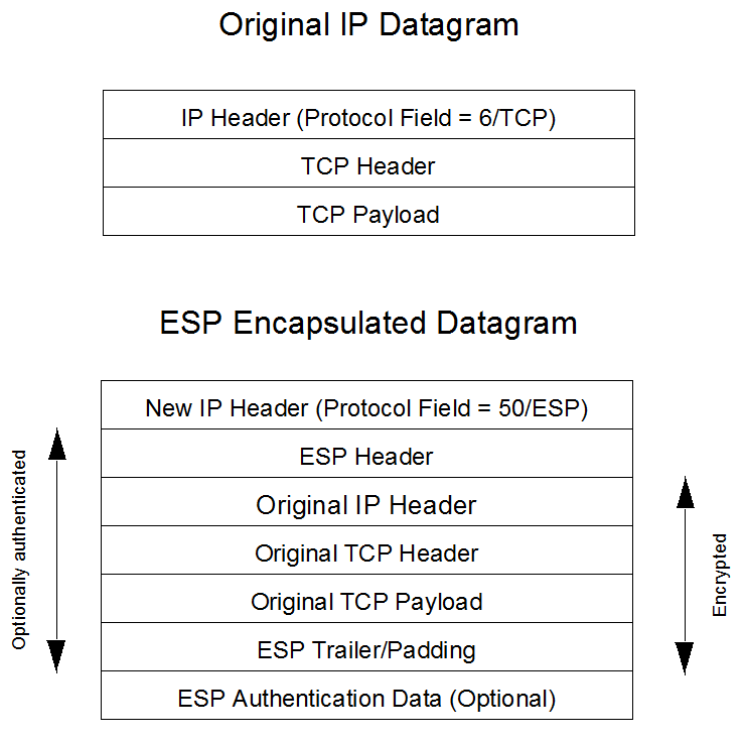
AH Transport Mode



AH Tunnel Mode



ESP Transport Mode



ESP Tunnel Mode

```
ORACLE(ike-sainfo)# security-protocol esp
ORACLE(ike-sainfo)#
```

4. Use the **auth-algo** parameter to specify the authentication algorithms supported by this SA.

   The following authentication protocols are available

Message Digest Algorithm 5 (md5) — as defined by RFC 1321, *The MD5 Message-Digest Algorithm*.

Secure Hash Algorithm (sha) — as defined by RFC 3174, *Secure Hash Standard*.

any (the default) — supports both MD5 and SHA-1.

```
ORACLE(ike-sainfo)# auth-algo md5
ORACLE(ike-sainfo)#
```

5. Use the **encryption-algo** parameter to specify the encryption algorithms supported by this SA.

   The following encryption protocols are available

   Triple DES (3des) — as defined by ANSI X.9.52 1998, *Triple Data Encryption Algorithm Modes of Operation*.

   Advanced Encryption Standard (aes) — as defined by RFC 3565, *Advanced Encryption Standard*.

   NULL Encryption (null) — as described in RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*. This option provides no privacy services, and is not recommended for production environments.

   any (the default) — supports all listed encryption protocols.

   ```
   ORACLE(ike-sainfo)# encryption-algo aes
   ORACLE(ike-sainfo)#
   ```

6. Use the **ipsec-mode** parameter to specify the IPSec operational mode.

   Transport mode (the default) provides a secure end-to-end connection between two IP hosts. Transport mode encapsulates the IP payload.

   Tunnel mode provides VPN service where entire IP packets are encapsulated within an outer IP envelope and delivered from source (an IP host) to destination (generally a secure gateway) across an untrusted internet.

   Refer to the previous figures for encapsulation details.

   ```
   ORACLE(ike-sainfo)# ipsec-mode tunnel
   ORACLE(ike-sainfo)#
   ```

7. If **ipsec-mode** is tunnel, use the required **tunnel-local-addr** parameter to specify the IP address of the local IKEv1 interface that terminates the IPsec tunnel.

   This parameter can safely be ignored if **ipsec-mode** is transport.

   ```
   ORACLE(ike-sainfo)# tunnel-local-addr 192.169.204.14
   ORACLE(ike-sainfo)#
   ```

8. If **ipsec-mode** is tunnel, use the **tunnel-remote-addr** parameter to specify the IP address of the remote IKEv1 peer that terminates the IPsec tunnel.

   Provide the remote IP address, or use the default wild-card value (*) to match all IP addresses.

   This parameter can safely be ignored if **ipsec-mode** is transport.

   ```
   ORACLE(ike-sainfo)# tunnel-remote-addr *
   ORACLE(ike-sainfo)#
   ```

9. Use **done**, **exit**, and **verify-config** to complete configuration of IKEv1 SA.
10. Repeat Steps 1 through 9 to configure additional IKEv1 SAs.

IPsec Security Policy Configuration

Use the following procedure to assign an IKEv1 SA to an existing IPsec Security Policy. Note that the network interface supported by the IPsec Security Policy must have been configured as an IKEv1 interface.

1. From superuser mode, use the following command sequence to access ike-config configuration mode. While in this mode, you configure global IKEv1 configuration parameters.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ipsec
   ORACLE(ipsec)# security-policy
   ORACLE(security-policy)#
   ```

2. Use the required **ike-sainfo-name** parameter to assign an IKv1 SA to this IPsec Security Policy.
3. Use **done**, **exit**, and **verify-config** to complete configuration of IPsec Security Policy.

SDP Session Description Protocol

The Secure Real-Time Transport Protocol, as described in RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*, provides a framework for the encryption and authentication of Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) streams. Both RTP and RTCP are defined by RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*.

Encryption ensures that the call content and associated signalling remains private during transmission. Authentication ensures that (1) received packets are from the purported source, (2) packets are not been tampered with during transmission, and (3) a packet has not been replayed by a malicious server.

Protocol Overview

While the RFC 3711 framework provides encryption and authentication procedures and defines a set of default cryptographic transforms required for RFC compliance, it does not specify a key management protocol to securely derive and exchange cryptographic keys. RFC4568, *Session Description Protocol (SDP) Security Description for Media Streams*, defines such a protocol specifically designed to exchange cryptographic material using a newly defined SDP crypto attribute. Cryptographic parameters are established with only a single message or in single round-trip exchange using the offer/answer model defined in RFC 3264, *An Offer/Answer Model with the Session Description Protocol*.

Release S-C6.2.0 provides support for an initial SDP Security Descriptions (SDES) implementation that generates keys used to encrypt SRTP/SRTCP packets. Authentication of packets will be added to a subsequent release.

A sample SDP exchange is shown below:

The SDP offerer sends:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
```

```
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

The SDP answerer replies:

```
v=0
o=sam 2890844526 2890842807 IN IP4 10.47.16.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 168.2.17.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20|1:4
```

The media-level SDP attribute, crypto, describes the cryptographic suite, key parameters, and session parameters for the preceding unicast media line. The crypto attribute takes the form:

a=crypto: tag crypto-suite key-parameter [session-parameters]

tag

The tag field contains a decimal number that identifies a specific attribute instance. When an offer contains multiple crypto attributes, the answer uses the tag value to identify the accepted offer.

In the sample offer the tag value is 1.

crypto-suite

The crypto-suite field contains the encryption and authentication algorithms, either AES_CM_128_HMAC_SHA1_80 orAES_CM_128_HMAC_SHA1_32.

The key-parameter field contains one or more sets of keying material for the selected crypto-suite and it has following format.

```
"inline:" <key||salt> ["|" lifetime] ["|" MKI ":" length]
```

inline is a method and specifies that the crypto material to be used by the offerer is transmitted via the SDP.

The key||salt field contains a base64-encoded concatenated master key and salt.

Assuming the offer is accepted, the key || salt provides the crypto material used by the offerer to encrypt SRTP/SRTCP packets, and used by the answerer to decrypt SRTP/SRTCP packets.

Conversely the key || salt contained in the answer to the offer provides the crypto material used by the answerer to encrypt SRTP/SRTCP packets, and used by the offerer to decrypt SRTP/SRTCP packets.

The lifetime field optionally contains the master key lifetime (maximum number of SRTP or SRTCP packets encoded using this master key).

In the sample offer the lifetime value is 1,048, 576 (220) packets.

The MKI:length field optionally contains the Master Key Index (MKI) value and the MKI length.

The MKI is used only when the offer contains multiple keys; it provides a means to differentiate one key from another. The MKI takes the form of an integer, followed by its byte length when included in SRTP/SRTCP packets.

In the sample offer the MKI value is 1 with a length of 4 bytes.

The session-parameters field contains a set of optional parameters that may override SRTP session defaults for the SRTP and SRTCP streams.

UNENCRYPTED_SRTP — SRTP messages are not encrypted

UNENCRYPTED_SRTCP — SRTCP messages are not encrypted

UNAUTHENTICATED_SRTP — SRTP messages are not authenticated

When generating an initial offer, the offerer must ensure that there is at least one crypto attribute for each media stream for which security is desired. Each crypto attribute for a given media stream must contain a unique tag. The ordering of multiple crypto attributes is significant — the most preferred crypto suite is listed first.

Upon receiving the initial offer, the answerer must either accept one of the offered crypto attributes, or reject the offer in its entirety.

When an offered crypto attribute is accepted, the crypto attribute in the answer MUST contain the tag and crypto-suite from the accepted crypto attribute in the offer, and the key(s) the answerer will be using for media sent to the offerer.

The crypto-suite is bidirectional and specifies encryption and authentication algorithms for both ends of the connection. The keys are unidirectional in that one key or key set encrypts and decrypts traffic originated by the offerer, while the other key or key set encrypts and decrypts traffic originated by the answerer. The use of symmetric keying, where the same key is used for both encryption and decryption, mandates the key exchange between the offerer and the answerer.

Key exchange via text-based SDP is unacceptable in that malicious network elements could easily eavesdrop and obtain the plaintext keys, thus compromising the privacy and integrity of the encrypted media stream. Consequently, the SDP exchange must be protected by a security protocol such as IPsec or TLS.

Licensing and Hardware Requirements

SRTP/SRTCP support requires the presence of an IPsec NIU and an SSM/SSM2 (Security Service Module).

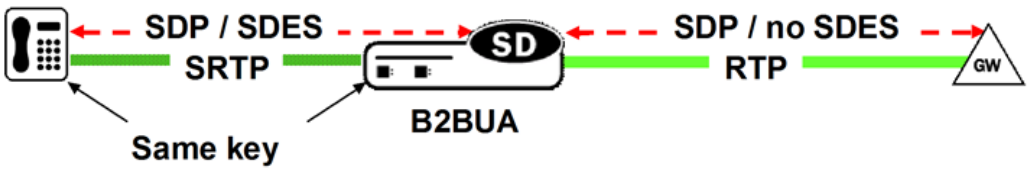
No additional licences are required.

Operational Modes

SRTP topologies can be reduced to three basic topologies which are described in the following sections.

Single-Ended SRTP Termination

Single-ended SRTP termination is illustrated in the following figure.



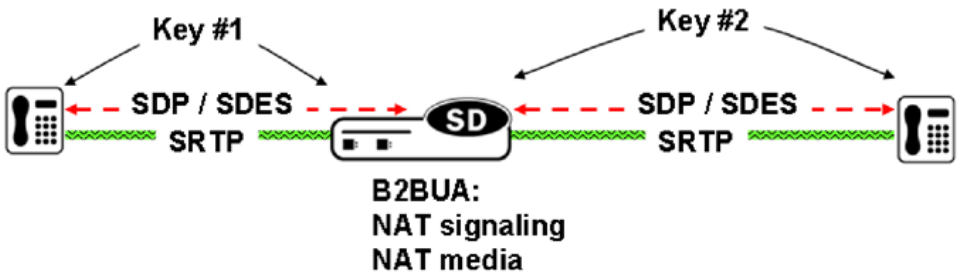
Single-Ended SRTP Termination

If SRTP is enabled for the inbound realm/interface, the Oracle USM handles the incoming call as specified by the Media Security Policy assigned to the inbound realm. If there is crypto attribute contained in the offer, the Oracle USM parses the crypto attributes and optional parameters, if any. If the offer contains a crypto attribute or attributes compatible with the requirements specified by the SDES profile assigned to the Media Security policy, it selects the most preferred compatible attribute. Otherwise, the Oracle USM rejects the offer. Before the SDP is forwarded to the called party, the Oracle USM allocates resources, established SRTP and SRTCP Security Associations and updates the SDP by removing the crypto attribute and inserting possibly NAT'ed media addresses and ports. At the same time, the original crypto attribute is also removed from the SDP.

Once the reply from the called party is received, the Oracle USM inserts appropriate crypto attribute(s) to form a new SDP, and forward the response back to the calling party.

Back-to-Back SRTP Termination

Back-to-back SRTP termination is illustrated in the following figure.

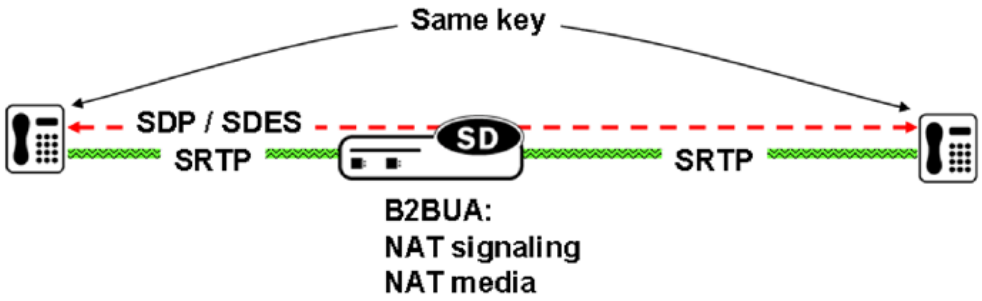


Back-to-Back SRTP Termination

Initial processing is similar to the single-ended termination described above. Before forwarding the request to the called party, the Oracle USM replaces the original crypto attribute with a new one whose crypto attribute conforms to the media security policy for the outbound realm/interface. Upon receiving the answer from the called party, the Oracle USM accepts or rejects it, again based upon conformity to the media security policy. If accepted, the Oracle USM replaces the original crypto attribute from the called party with its own to form a new SDP, which it forwards back to the calling party. At this point, SRTP media sessions are established on both sides for both calling and called parties.

SRTP Pass-Thru

SRTP pass-thru is illustrated in the following figure.



SRTP Pass-Thru

If the media security policy specifies pass-through mode, the Oracle USM does not alter the crypto attribute exchange between the calling and the called party; the attribute is transparently passed.

SDES Configuration

SDES configuration consists of the following steps.

1. Create one or more SDES profiles which specify parameter values negotiated during the offer/answer exchange.
2. Create one or more Media Security Policies that specify key exchange protocols and protocol-specific profiles.
3. Assign a Media Security Policy to a realm.
4. Create an interface-specific Security Policy.

SDES Profile Configuration

An SDES profile specifies the parameter values offered or accepted during SDES negotiation.

To configure SDES profile parameters:

1. From superuser mode, use the following command sequence to access sdes-profile configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Use the required **name** parameter to provide a unique identifier for this sdes-profile instance.

   name enables the creation of multiple sdes-profile instances.

3. Use the **crypto-suite** parameter to select the encryption and authentication algorithms accepted or offered by this sdes-profile.

> **Note:**
>
> SRTP authentication is not currently supported.

Allowable values are:

AES_CM_128_HMAC_SHA1_80 (the default value)

supports AES/128 bit key for encryption and HMAC/SHA-1 80-bit digest for authentication

AES_CM_128_HMAC_SHA1_32

supports AES/128 bit key for encryption and HMAC/SHA-1 32-bit digest for authentication

4. Because SRTP authentication is not currently supported, ignore the **srtp-auth** parameter.
5. Use the **srtp-encrypt** parameter to enable or disable the encryption of RTP packets.

   With encryption enabled, the default condition, the Oracle USM offers RTP encryption, and rejects an answer that contains an UNENCRYPTED_SRTP session parameter in the crypto attribute.

   With encryption disabled, the Oracle USM does not offer RTP encryption and includes an UNENCRYPTED_SRTP session parameter in the SDP crypto attribute; it accepts an answer that contains an UNENCRYPTED_SRTP session parameter.
6. Use the **srtcp-encrypt** parameter to enable or disable the encryption of RTCP packets.

   With encryption enabled, the default condition, the Oracle USM offers RTCP encryption, and rejects an answer that contains an UNENCRYPTED_SRTCP session parameter in the crypto attribute.

   With encryption disabled, the Oracle USM does not offer RTCP encryption and includes an UNENCRYPTED_SRTCP session parameter in the SDP crypto attribute; it accepts an answer that contains an UNENCRYPTED_SRTCP session parameter.
7. Use the **mki** parameter to enable or disable the inclusion of the MKI:length field in the SDP crypto attribute.

   The master key identifier (MKI) is an optional field within the SDP crypto attribute that differentiates one key from another. MKI is expressed as a pair of decimal numbers in the form: |mki:mki_length| where mki is the MKI integer value and mki_length is the length of the MKI field in bytes.

   The MKI field is necessary only in topologies that may offer multiple keys within the crypto attribute.

   Allowable values are enabled and disabled (the default).

   enabled – an MKI field is sent within the crypto attribute (16 bytes maximum)

   disabled – no MKI field is sent
8. Use **done**, **exit**, and **verify-config** to complete configuration of this SDES profile instance.
9. Repeat Steps 1 through 8 to configure additional SDES profiles.

Media Security Policy Configuration

Use the following procedure to create a Media Security Policy that specifies the role of the Oracle USM in the security negotiation. If the Oracle USM takes part in the negotiation, the policy specifies a key exchange protocol and SDES profile for both incoming and outgoing calls.

To configure media-security-policy parameters:

1. From superuser mode, use the following command sequence to access media-sec-policy configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# media-security
   ORACLE(media-security)# media-sec-policy
   ORACLE(media-sec-policy)#
   ```

2. Use the required **name** parameter to provide a unique identifier for this media-sec-policy instance.

   name enables the creation of multiple media-sec-policy instances.

3. Use optional **pass-thru** parameter to enable or disable pass-thru mode.

   With pass-thru mode enabled, the User Agent (UA) endpoints negotiate security parameters between each other; consequently, the Oracle USM simply passes SRTP traffic between the two endpoints.

   With pass-thru mode disabled (the default state), the Oracle USM disallows end-to-end negotiation — rather the Oracle USM initiates and terminates SRTP tunnels with both endpoints.

4. Use the **outbound** navigation command to move to media-sec-outbound configuration mode. While in this configuration mode you specify security parameters applied to the outbound call leg, that is calls sent by the Oracle USM.

5. Use the **protocol** parameter to select the key exchange protocol.

   Select sdes for SDES key exchange.

6. Use the **profile** parameter to specify the name of the SDES profile applied to calls sent by the Oracle USM.

7. Use the **mode** parameter to select the real time transport protocol.

   Allowable values are rtp and srtp (the default).

   mode identifies the transport protocol (RTP or SRTP) included in an SDP offer when this media-security-policy is in effect.

8. Use the **done** and **exit** parameters to return to media-sec-policy configuration mode.

9. Use the **inbound** navigation command to move to media-sec-inbound configuration mode. While in this configuration mode you specify security parameters applied to the inbound call leg, that is calls received by the Oracle USM.

10. Use the **protocol** parameter to select the key exchange protocol.

    Select sdes for SDES.

11. Use the **profile** parameter to specify the name of the SDES profile applied to calls received by the Oracle USM.

12. Use the **mode** parameter to select the real time transport protocol.

    Allowable values are rtp and srtp (the default).

    mode identifies the transport protocol (RTP or SRTP) accepted in an SDP offer when this media-security-policy is in effect.

13. Use **done**, **exit**, and **verify-config** to complete configuration of this media security policy instance.

14. Repeat Steps 1 through 13 to configure additional media-security policies.

Assign the Media Security Policy to a Realm

To assign a media-security-policy to a realm:

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in this mode, you assign an existing media-security-policy to an existing realm.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
identifier:
1. access-12
...
...
selection: 1
ORACLE(realm-config)#
```

2. Use the **media-sec-policy** parameter to assign the policy to the target realm.
3. Use **done**, **exit**, and **verify-config** to complete assignment of the media-security-policy to the realm.

ACLI Example Configurations

The following section contain relevant sections of system configurations for basic operational modes.

Single-Ended SRTP Termination Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
    name                    sdes1
    crypto-list             AES_CM_128_HMAC_SHA1_80
    srtp-auth               enabled
    srtp-encrypt            enabled
    srtcp-encrypt           enabled
    mki                     disabled
    key
    salt
    last-modified-by        admin@console
    last-modified-date      2009-11-16 15:37:13
media-sec-policy
    name                    msp2
    pass-through            disabled
    inbound
        profile             sdes1
        mode                srtp
        protocol            sdes
    outbound
        profile             sdes1
        mode                srtp
        protocol            sdes
    last-modified-by        admin@console
    last-modified-date      2009-11-16 15:37:51
```

```
...
...
...
realm-config
    identifier                  peer
    description
    addr-prefix                 192.168.0.0/16
    network-interfaces          M00:0
    mm-in-realm                 enabled
    mm-in-network               enabled
    mm-same-ip                  enabled
    mm-in-system                enabled
    bw-cac-non-mm               disabled
    msm-release                 disabled
    qos-enable                  disabled
    generate-UDP-checksum       disabled
    max-bandwidth               0
    fallback-bandwidth          0
    max-priority-bandwidth      0
    max-latency                 0
    max-jitter                  0
    max-packet-loss             0
    observ-window-size          0
    parent-realm
    dns-realm
    media-policy
    media-sec-policy            msp2
    in-translationid
...
...
...
    last-modified-by            admin@console
    last-modified-date          2009-11-10 15:38:19
```

Back-to-Back SRTP Termination Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
   name                         sdes1
   crypto-list                  AES_CM_128_HMAC_SHA1_80
   srtp-auth                    enabled
   srtp-encrypt                 enabled
   srtcp-encrypt                enabled
   mki                          disabled
   key
   salt
   last-modified-by             admin@console
   last-modified-date           2009-11-16 15:37:13
media-sec-policy
   name                         msp2
   pass-through                 disabled
   inbound
      profile                   sdes1
      mode                      srtp
      protocol                  sdes
   outbound
      profile                   sdes1
      mode                      srtp
```

```
      protocol                  sdes
   last-modified-by          admin@console
   last-modified-date        2009-11-16 15:37:51
...
...
...
realm-config
   identifier                peer
   description
   addr-prefix               192.168.0.0/16
   network-interfaces        M00:0
   mm-in-realm               enabled
   mm-in-network             enabled
mm-same-ip                   enabled
   mm-in-system              enabled
   bw-cac-non-mm             disabled
   msm-release               disabled
   qos-enable                disabled
   generate-UDP-checksum     disabled
   max-bandwidth             0
   fallback-bandwidth        0
   max-priority-bandwidth    0
   max-latency               0
   max-jitter                0
   max-packet-loss           0
   observ-window-size        0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy    msp2
...
...
...
realm-config
   identifier                core
   description
   addr-prefix               172.16.0.0/16
   network-interfaces        M10:0
   mm-in-realm               enabled
   mm-in-network             enabled
   mm-same-ip                enabled
   mm-in-system              enabled
   bw-cac-non-mm             disabled
   msm-release               disabled
   qos-enable                disabled
   generate-UDP-checksum     disabled
   max-bandwidth             0
   fallback-bandwidth        0
   max-priority-bandwidth    0
   max-latency               0
   max-jitter                0
   max-packet-loss           0
   observ-window-size        0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy          msp2
   in-translationid
...
...
...
```

```
   last-modified-by           admin@console
   last-modified-date         2009-11-10 15:38:19
```

SRTP Pass-Thru Configuration

```
ORACLE# show running-config
...
...
...
sdes-profile
   name                              sdes1
   crypto-list                       AES_CM_128_HMAC_SHA1_80
   srtp-auth                         enabled
   srtp-encrypt                      enabled
   srtcp-encrypt                     enabled
   mki                               disabled
   key
   salt
   last-modified-by                  admin@console
   last-modified-date                2009-11-16 15:37:13
media-sec-policy
   name                              msp2
   pass-through                      enabled
   inbound
      profile                        sdes1
      mode                           srtp
      protocol                       sdes
   outbound
      profile                        sdes1
      mode                           srtp
      protocol                       sdes
   last-modified-by                  admin@console
   last-modified-date                2009-11-16 15:37:51
...
...
...
realm-config
   identifier                           peer
   description
   addr-prefix                          192.168.0.0/16
   network-interfaces                M00:0
   mm-in-realm                          enabled
   mm-in-network                        enabled
   mm-same-ip                           enabled
   mm-in-system                         enabled
   bw-cac-non-mm                        disabled
   msm-release                          disabled
   qos-enable                           disabled
   generate-UDP-checksum                disabled
   max-bandwidth                        0
   fallback-bandwidth                   0
   max-priority-bandwidth               0
   max-latency                          0
   max-jitter                           0
   max-packet-loss                      0
   observ-window-size                   0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy                   msp2
...
```

```
...
...
realm-config
   identifier                          core
   description
   addr-prefix                         172.16.0.0/16
   network-interfaces                M10:0
   mm-in-realm                         enabled
   mm-in-network                       enabled
   mm-same-ip                          enabled
   mm-in-system                        enabled
   bw-cac-non-mm                       disabled
   msm-release                         disabled
   qos-enable                          disabled
   generate-UDP-checksum               disabled
   max-bandwidth                       0
   fallback-bandwidth                  0
   max-priority-bandwidth              0
   max-latency                         0
   max-jitter                          0
   max-packet-loss                     0
   observ-window-size                  0
   parent-realm
   dns-realm
   media-policy
   media-sec-policy                   msp2
   in-translationid
   ...
   ...
   ...
   last-modified-by                   admin@console
   last-modified-date                 2009-11-10 15:38:19
```

Security Policy

A Security Policy enables the Oracle USM to identify inbound and outbound media streams that are treated as SRTP/SRTCP. The high-priority Security Policy, p1, (shown below) allows signaling traffic from source 172.16.1.3 to destination 172.16.1.10:5060. The lower-priority Security Policy, p2, (also shown below) matches media traffic with the same source and destination, but without any specific ports. Consequently, SIP signaling traffic (from local port 5060) go through, but the media stream will be handled by appropriate SRTP SA.

```
security-policy
     name                              p1
     network-interface                 private:0
     priority                          0
     local-ip-addr-match               172.16.1.3
     remote-ip-addr-match              172.16.1.10
     local-port-match                  5060
     remote-port-match                 0
     trans-protocol-match              UDP
     direction                         both
     local-ip-mask                     255.255.255.255
     remote-ip-mask                    255.255.255.255
     action                            allow
     ike-sainfo-name
     outbound-sa-fine-grained-mask
          local-ip-mask                255.255.255.255
          remote-ip-mask               255.255.255.255
          local-port-mask              0
```

```
            remote-port-mask              0
            trans-protocol-mask           0
            valid                         enabled
            vlan-mask                     0xFFF
       last-modified-by                   admin@console
       last-modified-date                 2009-11-09 15:01:55

security-policy
       name                               p2
       network-interface                  private:0
       priority                           10
       local-ip-addr-match                172.16.1.3
       remote-ip-addr-match               172.16.1.10
       local-port-match                   0
       remote-port-match                  0
       trans-protocol-match               UDP
       direction                          both
       local-ip-mask                      255.255.255.255
       remote-ip-mask                     255.255.255.255
       action                             srtp
       ike-sainfo-name
       outbound-sa-fine-grained-mask
            local-ip-mask                 0.0.0.0
            remote-ip-mask                255.255.255.255
            local-port-mask               0
            remote-port-mask              65535
            trans-protocol-mask           255
            valid                         enabled
            vlan-mask                     0xFFF
       last-modified-by                   admin@console
       last-modified-date                 2009-11-09 15:38:19
```

Modified ALCI Configuration Elements

The action parameter in security-policy configuration mode has been modified to accept additional values, srtp and srtcp.

1. From superuser mode, use the following command sequence to access media-sec-policy configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# ipsec
   ORACLE(ipsec)# security-policy
   ORACLE(security-policy)# action ?
   <enumeration> action (default: ipsec)
   ipsec, allow, discard, srtp, srtcp
   ORACLE(security-policy)#
   ```

   The **show security** command has been updated with an **srtp** option.

   ```
   ORACLE# show security srtp
   sad
   spd
   statistics
   SRTP Statistics
   status
   ```

   The **srtp** option is similar to the **ipsec** option save for the **sad** sub-option that provides data for only SRTP SAs.

The **show sa stats** command has been updated with an **srtp** option.

```
ORACLE# show sa stats
<ENTER>    Show statistics summary of all Security Associations
<ike>    Show statistics for IKE Security Associations
<ims-aka>    Show statistics for IMS-AKA Security Associations
<srtp>    Show statistics for SRTP Security Associations
sd# show sa stats srtp
20:06:24-114
SA Statistics                    ---- Lifetime ----
                           Recent   Total   PerMax
SRTP Statistics
ADD-SA Req Rcvd              0        0        0
ADD-SA Success Resp Sent     0        0        0
ADD-SA Fail Resp Sent        0        0        0
DEL-SA Req Rcvd              0        0        0
DEL-SA Success Resp Sent     0        0        0
DEL-SA Fail Resp Sent        0        0        0
SA Added                     0        0        0
SA Add Failed                0        0        0
SA Deleted                   0        0        0
SA Delete Failed             0        0        0
```

Secure and Non-Secure Flows in the Same Realm

To simplify deployments, the Oracle USM allows secure and non-secure flows in the same realm.This broadened set of capabilities means the Oracle USM can support RTP and SRTP flows, and it can support a larger group of UAs that might have varying SRTP abilities. Prior to this release, when a cryptographic session arrived at the Oracle USM and failed to match an applicable media security profile, it was rejected.

This broadened support for secure and non-secure flows and for UAs with various SRTP abilities is established throughout the system, residing in these configurations:

- media-sec-policy
- sdes-profile

While configurations reside there, you should also note special considerations for the **security-policy** configuration and implications for security associations.

Mode Settings in the Media Security Policy

The media security policy configuration's **mode** parameter offers three settings. It is the **any** mode that allows you to support secure and non-secure flows in the same realm.

For Incoming Flows

This section describes the way all three settings behavior for incoming flows.

- **rtp**—If the incoming media security policy associated with a realm has **rtp** set as its mode, then the Oracle USM only accepts offer SDP containing RTP/AVP media lines. Otherwise, the Oracle USM rejects the session with a 488 Not Acceptable Here.
- **srtp**—If the incoming media security policy associated with a realm has s**rtp** set as its mode, the Oracle USM only accepts offer SDP containing RTP/SAVP media lines. Otherwise, the Oracle USM rejects the session with a 488 Not Acceptable Here.

- **any**—If the incoming media security policy associated with a realm has **any** set as its mode, the Oracle USM accepts offer SDP that has RTP/AVP media lines, RTP/SAVP media lines, or both.

For Outgoing Flows

This section describes the way all three settings behavior for outgoing flows.

- **rtp**—If the outgoing media security policy associated with a realm has **rtp** set as its mode, then the Oracle USM converts any RTP/SAVP media lines from incoming offer SDP to RTP/AVP for the offer SDP it sends out.
  Incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80 inline:f0oLKTuMYwXqrKa7Ch
+MOBvLe8YnXnD6Kmnj4LQ2
```

  The Oracle USM will take that and convert it to the following for outgoing traffic.

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 6000 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
```

  This conversion can result in multiple media lines with RTP/AVP for the same media profile and an RTP/SAVP media line for the same media profile. To prevent duplicate lines in the SDP the Oracle USM sends, the Oracle USM inspects incoming SDP to determine is RTP/AVP and RTP/SAVP media lines exist for the same media profile. If it finds such a media profile, the Oracle USM disables the RTP/AVP (by setting the port to 0 in the outgoing offer SDP) corresponding to the RTP/AVP media line for that media profile. Doing so forces the UA answering the SDP offer to choose the media lines corresponding to the RTP/SAVP media lines in the incoming offer SDP. An SRTP-RTP session results.

  The incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
```

```
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5012 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80 inline:f0oLKTuMYwXqrKa7Ch
+MOBvLe8YnXnD6Kmnj4LQ2
```

And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 0 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 6002 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
```

- **srtp**—If the outgoing media security policy associated with a realm has s**rtp** set as its mode, the Oracle USM converts any RTP/AVP media lines from an incoming offer SDP to RTP/SAVP for the offer SDP the Oracle USM sends.
  The incoming offer SDP might look like this:

  ```
  v=0
  o=MxSIP 0 1480968866 IN IP4 192.168.22.180
  s=SIP Call
  c=IN IP4 192.168.22.180
  t=0 0
  m=audio 5012 RTP/AVP 0 8 18 0 101
  a=rtpmap:0 PCMU/8000
  ```

```
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
```

And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 6000 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
a=crypto:0 AES_CM_128_HMAC_SHA1_80 inline:f0oLKTuMYwXqrKa7Ch
+MOBvLe8YnXnD6Kmnj4LQ2
```

This conversion might result in multiple media lines with RTP/SAVP for the same media profile if the incoming offer SDP has an RTP/AVP media line and an RTP/SAVP media for the same media profile. To prevent multiple identical media lines in the SDP it sends, the Oracle USM inspects the incoming SDP to determine whether both RTP/AVP and RTP/SAVP media lines exist for the same media profile. If it finds such a media profile, the Oracle USM disables the RTP/SAVP (by setting the port to 0 in the outgoing offer SDP) corresponding to the RTP/AVP media line for that media profile. Doing so forces the UA answering the SDP offer to choose the media lines corresponding to the RTP/SAVP media lines in the incoming offer SDP. An SRTP-SRTP session results.

The incoming offer SDP might look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 192.168.22.180
s=SIP Call
c=IN IP4 192.168.22.180
t=0 0
m=audio 5012 RTP/AVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 5010 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
```

```
a=fmtp:101 0-15
a=crypto:0 AES_CM_128_HMAC_SHA1_80 inline:f0oLKTuMYwXqrKa7Ch
+MOBvLe8YnXnD6Kmnj4LQ2
```

And the outgoing offer SDP will look like this:

```
v=0
o=MxSIP 0 1480968866 IN IP4 172.16.22.180
s=SIP Call
c=IN IP4 172.16.22.180
t=0 0
m=audio 0 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-15
m=audio 6002 RTP/SAVP 0 8 18 0 101
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=silenceSupp:off - - - -
a=fmtp:18 annexb=no
a=fmtp:101 0-1
a=crypto:0 AES_CM_128_HMAC_SHA1_80 inline:f0oLKTuMYwXqrKa7Ch
+MOBvLe8YnXnD6Kmnj4LQ2
```

- **any**—If the outgoing media security policy associated with a realm has **any** set as its mode, the Oracle USM creates offer SDP based on the value configured in the **egress-offer-format**, which can be set in the **sdes-profile** configuration.
  - If the value is **same-as-ingress**, the Oracle USM leaves the profile of the media lines unchanged.
  - If the value is **simultaneous-best-effort**, the Oracle USM inspects the incoming offer SDP and:
    * Adds an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile
    * Adds an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile
      Should the media profile in the incoming offer SDP already have two media lines (one fore RTP/AVP and on for RTP/SAVP), the Oracle USM does not have to make these additions. It will map the media lines in the answer it receives with the media lines from the incoming offer SDP. It will also ensure that media lines in the answer SDP it sends match the media lines from the incoming offer SDP.

Security Associations for RTP and RTCP

With RTP and SRTP supported in the same realm, you want to configure your SRTP security policies to preserve system resources and exercise the full capability number of licensed sessions. You need to do to avoid session agent interaction that can have an adverse impact on the number of sessions.

To do so, check the **local-ip-match-address** for the STRP security policy has an IP address different from the all steering pool IP addresses for realms requiring both RTP and SRTP. The Oracle USM recognizes this difference automatically and sets the connection address of media lines in SDP accordingly:

- The connection address for RTP media lines is the IP address of the applicable steering pool. The Oracle USM passes through RTP and RTCP packets sent by and received from the steering pool IP address. This operation requires no reference to session agents because the steering pool address does not match the IP address for the SRTP security policy's **local-ip-address-match** value.
- The connection address of the SRTP media lines continues to be the **local-ip-address-match** value from the applicable SRTP security policy.

Since RTP and RTCP packets are sent to and from the steering pool's IP address (an IP address for which there is no SRTP security policy configured), there is no reason to reference session agents.

> **Note:**
>
> Oracle's Enhanced Traffic Controller (ETC) networking interface unit handles traffic differently such the issue with session agent reference is elided. That is, if you are using the ETC NIU, you do not need to be concerned about this issue.

Security Configuration for RTP and RTCP

This section shows you how to configure your Oracle USM to support secure and non-secure flows in the same realm.

To configure a security policy to support secure and non-secure flows in the same realm:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **media-security** and press Enter.

   ```
   ORACLE(security)# media-security
   ORACLE(media-security)#
   ```

4. Type **media-sec-policy** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

   ```
   ORACLE(media-security)# media-sec-policy
   ORACLE(media-sec-policy)#
   ```

5. Type **inbound** to enter the setting for inbound flows.

   ```
   ORACLE(media-sec-policy)# inbound
   ORACLE(inbound)#
   ```

6. **mode**—Enter the mode that you want to use for inbound flows. You can choose from **rtp**, **srtp**, and **any**.

7. **protocol**—Change this value from **sdes** to **none**. Use the **done** command to save your work, and exit the **inbound** configuration.
8. Type **outbound** to enter the setting for inbound flows.

   ```
   ORACLE(media-sec-policy)# outbound
   ORACLE(outbound)#
   ```

9. **mode**—Enter the mode that you want to use for outbound flows. You can choose from **rtp**, **srtp**, and **any**.
10. **protocol**—Change this value from **sdes** to **none**. Use the **done** command to save your work, and exit the **outbound** configuration.
11. Type **done** and continue.

    To set the egress offer format for an SDES profile configuration:

12. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ORACLE(configure)#
    ```

13. Type **security** and press Enter.

    ```
    ORACLE(configure)# security
    ORACLE(security)#
    ```

14. Type **media-security** and press Enter.

    ```
    ORACLE(security)# media-security
    ORACLE(media-security)#
    ```

15. Type **sdes-profile** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

    ```
    ORACLE(media-security)# sdes-profile
    ORACLE(sdes-profile)#
    ```

16. **egress-offer-format—Choose an egress offer format for this profile to use when you set the outbound mode in the media security policy to any. You can select one of two values:**
    - If the value is **same-as-ingress (default)**, the Oracle USM leaves the profile of the media lines unchanged.
    - If the value is **simultaneous-best-effort**, the Oracle USM inspects the incoming offer SDP and:
      – Adds an RTP/SAVP media line for any media profile that has only the RTP/AVP media profile
      – Adds an RTP/AVP media line for any media profile that has only the RTP/SAVP media profile
17. Type **done** to save your work and continue.

Supporting UAs with Different SRTP Capabilities

To support UAs with different levels of SRTP capabilities, the **use-ingress-session-params** parameter appears in the **sdes-profile** configuration. The values for this parameter allow the Oracle USM to accept and (where applicable) mirror the UA's proposed cryptographic session parameters:

- **srtp-auth**—Decides whether or not authentication is performed in SRTP

- **srtp-encrypt**—Decides whether or not encryption is performed in SRTP
- **srtcp-encrypt**—Decides whether or not encryption is performed in SRTCP

Using these possible values, the Oracle USM accepts the corresponding incoming session parameters.

Receiving Offer SDP

When the Oracle USM receives offer SDP with applicable session parameters, it uses the same session parameters in its answer SDP (if it can support the same). This is true even if the value for that session parameter differs from the available media security profile.

Consider this example: An SDES profile is applied for incoming direction for a media security policy configured with the **srtcp-encrypt** value set to **enabled**. With the **use-ingress-session-params** parameter set to **srtcp-encrypt** for the SDES profile, the Oracle USM accepts the offer SDP and also sets UNENCRYPTED_SRTCP for the cryptographic attributes in its answer SDP. When the call connects, the Oracle USM does not encrypt or decrypt SRTCP packets. Without the SDES profile set this way, the Oracle USM would reject offer SDP if any of its cryptographic attributes showed UNENCRYPTED_SRTCP in its session parameters list.

Receiving Answer SDP

When the Oracle USM receives answer SDP with the accepted session parameter, the value for the same session parameters that the Oracle USM uses might or might not be the same as the incoming value. Configuration of the outbound media security profile controls the value used because the Oracle USM makes offer SDP, which cannot be changed, with the session parameters based on the outgoing media security profile.

Consider this example: An SDES profile is applied for incoming direction for a media security policy configured with the **srtcp-encrypt** value set to **enabled**, so the cryptographic attributes in the SDP the system sends **do not have the UNENCRYPTED_SRTCP** session parameters. If the UNENCRYPTED_SRTCP appears in the corresponding answer SDP it receives, the Oracle USM accepts it if the **srtcp-encrypt** value appears in the **use-ingress-session-params** parameter. But the Oracle USM still performs SRTCP encryption. When the call connects, the Oracle USM encrypts outgoing SRTCP packets but does not decrypt incoming SRTCP packets. So if the UA (receiving the system's offer SDP) does not support SRTCP decryption, it will likely reject the offer SDP.

SDES Profile Configuration

To set the ingress session parameters for an SDES profile configuration:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **media-security** and press Enter.

   ```
   ORACLE(security)# media-security
   ORACLE(media-security)#
   ```

4. Type **sdes-profile** and press Enter. If you are editing a pre-existing configuration, you needs to select it before you can make changes.

```
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

5. **use-ingress-session-params—Enter the list of** values for which the Oracle USM will accept and (where applicable) mirror the UA's proposed cryptographic session parameters:
   - **srtp-auth**—Decides whether or not authentication is performed in SRTP
   - **srtp-encrypt**—Decides whether or not encryption is performed in SRTP
   - **srtcp-encrypt**—Decides whether or not encryption is performed in SRTCP

   ```
   ACMEPACKET(sdes-profile)# use-ingress-session-params srtp-auth srtp-
   encrypt srtcp-encrypt
   ```
6. Type **done** to save your work and continue.

Refining Interoperability

To refine any remaining interoperability issues, you can use the options parameter in the **media-sec-policy** and **sdes-profile** configurations.

Common values to configure an option are **include-local-id** and **include-remote-id**.

- **include-local-id**—The Oracle USM includes the IDi in the I_MESSAGE and the IDr in the R_MESSAGE.
  When used for the outbound direction of a media security policy, the IDi is included in the I_MESSAGE the Oracle USM sends. The content of the IDi is the value of the Contact header found in the INVITE message.
- **include-remote-id**—The system includes the IDr in the I_MESSAGE.

Multi-system Selective SRTP Pass-through

Prior to Release S-CX6.3F1, Oracle USM provided a single Secure Real-time Transport Protocol (SRTP) operational mode, referred to as SRTP Media Proxy Mode. In this original processing mode, each Oracle USM in the SRTP media path served as a proxy for the media — always decrypting inbound traffic, and encrypting outbound traffic.

Release S-CX6.3F1 introduces support for a new, alternative processing mode, referred to as Multi-system Selective SRTP Pass-through Mode. In this new mode encryption and decryption of SRTP media is handled by the SRTP endpoints, the calling and called parties. Off-load of the processor-intensive encryption/decryption provides the Oracle USM with the ability to handle a larger number of simultaneous SRTP sessions.

With Multi-system Selective SRTP Pass-through enabled, the Oracle USM can configured to selectively allow hair-pinned (spiral) SRTP packets to pass through the Oracle USM without encryption or decryption.

> **Note:**
>
> hair-pinned calls are those calls where the calling and called parties are within the same realm and/or within the same sub-network.

License Requirements

Multi-system Selective SRTP Pass-through requires only the SIP license.

Hardware Requirements

On the Acme Packet 4600 and Acme Packet 4500 platforms, Multi-system Selective SRTP Pass-through requires an SSM2 (for SIP/TLS signalling), and either an IPsec/SRTP NIU, or an ETC NIU.

Constraints

Multi-system Selective SRTP Pass-through support has the following constraints:

1. The realm, or realms in which the calling and called parties are located are configured for Multi-system Selective SRTP Pass-through support.
2. The call session does not require SIP—INFO to RFC 2833 tone translation.
3. The call session does not require SDES and can be used for the exchange of SRTP keying material. Both the calling and called parties must support the same key exchange protocol.
4. Multi-system Selective SRTP Pass-through support should not be enabled in topologies where core-side application servers may change the c-line to inject a media server or some other media device in the media path. In such cases, SRTP should be terminated at the SD for each endpoint, so that the media server receives unencrypted media. In the other case, where the c-line is not subject to modification, Multi-system Selective SRTP Pass-through can be enabled.

   If any of these conditions are not met, Multi-system Selective SRTP Pass-through processing cannot be provided, and the call is serviced as specified by SRTP Media Proxy Mode.

Operational Overview

To set up Multi-system Selective SRTP Pass-through, the ingress and egress Oracle USMs (which can, in fact, be a single Oracle USM) exchange the SDES keying material that they receive from their respective endpoint so that the Oracle USM peer can pass the material to its adjacent endpoint. The endpoint to endpoint exchange of keying material enables the endpoints themselves to generate encryption/decryption keys.

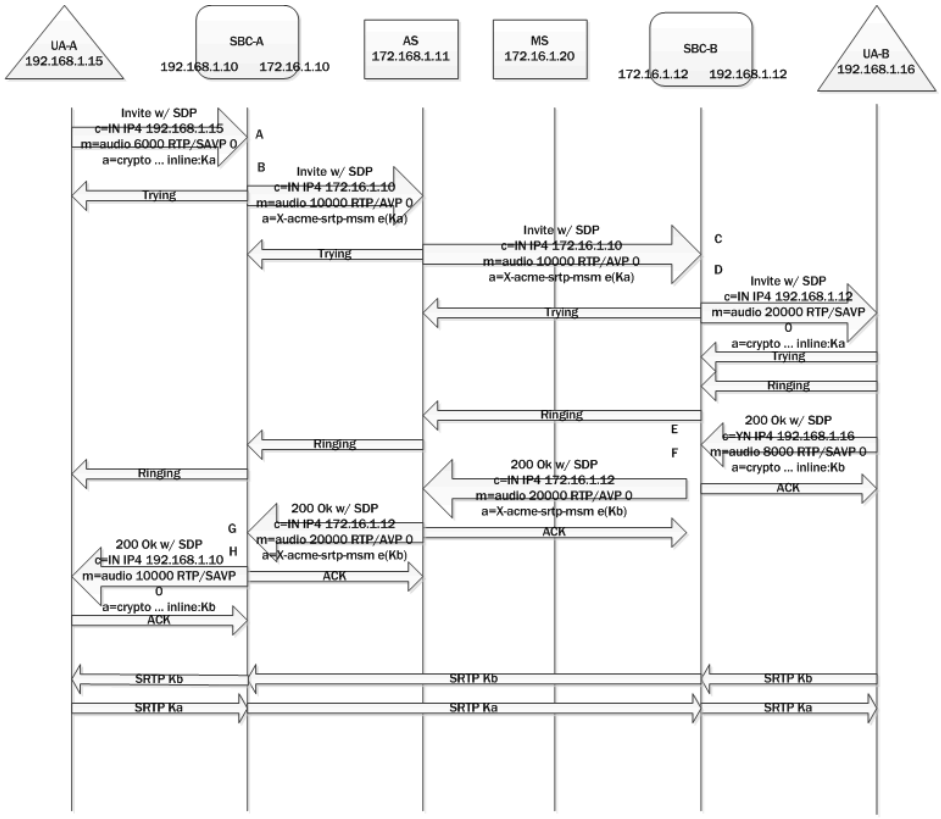
The actual exchange of keying material takes place in SIP messages (specifically, INVITE, 200 OK, and ACK) that carry offer or answer SDPs. Encrypted keying material is conveyed within a media attribute for each SRTP session. The name of the media attribute is configurable.

When either Oracle USM receives the encrypted keying material sent by its remote peer, it decrypts the media attribute and passes the plaintext attribute to its endpoint. Consequently, subsequent SRTP packets from the endpoints pass through the Oracle USMs without being decrypted and encrypted because both endpoints (now is possession of each others keying material) are able to decrypt the SRTP packets received from the other.

Call Flows

The following three sections describe call signalling for common scenarios.

Call Setup



192.168.1.15

A: The calling party sends an INVITE with SDP to SBC-A. The offer SDP contains an SDES crypto attribute within the SRTP media line.

B: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SD-A adds an a=X-acme-srtp-msm media attribute. The a=X-acme-srtp-msm attribute contains a cookie that includes an encryption of the SDES crypto attribute present in the SDP. The encryption is done using the shared secret configured for encrypting SRTP Pass-through information.

C: SBC-B receives the offer SDP that has the cookie sent by SBC-A. It is assumed that the proxies that forward the offer SDP sent by SBC-A preserve and forward the cookie added by SBC-A.

D: SBC-B checks if the egress realm has Multi-system Selective SRTP Pass-through enabled. If so, SBC-B decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-B adds the SDES crypto attribute retrieved from the cookie to the offer SDP sent to UA-B.

E: The called party sends an answer SDP with a SDES crypto attribute on the SRTP media line to SBC-B.

F: SBC-B checks if it has received a cookie in the offer SDP and adds the cookie to the answer SDP. The cookie contains an encryption of the SDES crypto attribute received in the answer

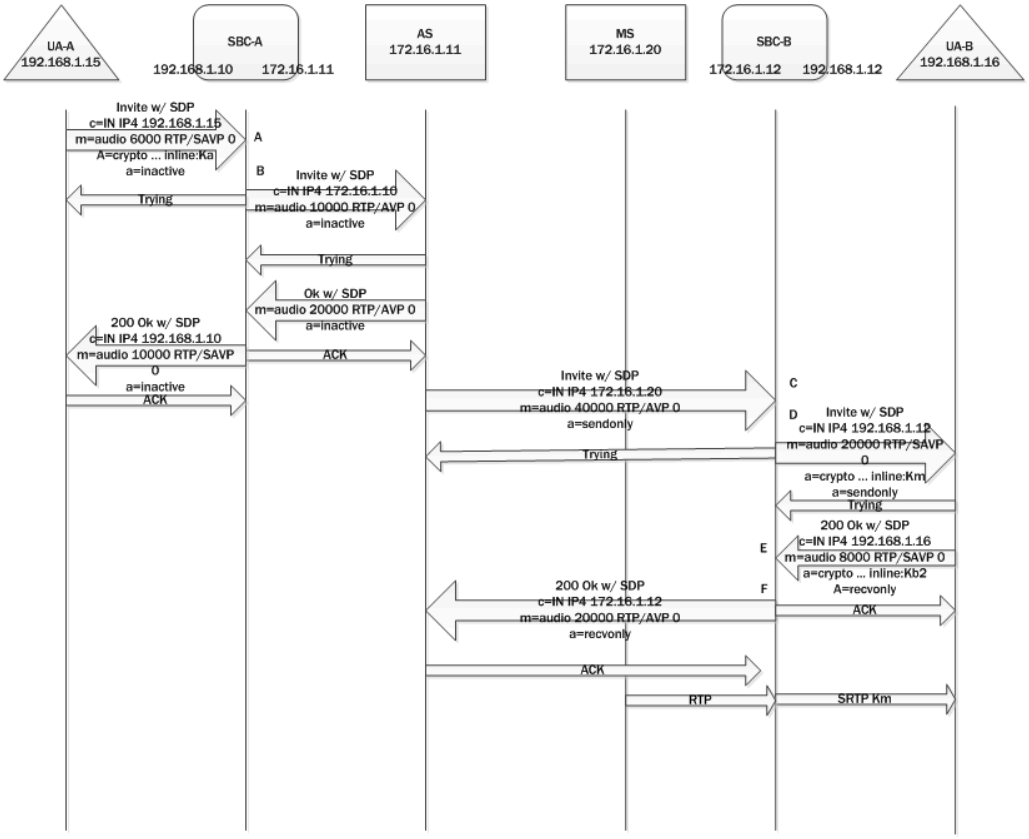
SDP from UA-B. SBC-B does not install any SA or media policy so that SRTP packets from/to UA-B can pass through SBC-B without any decryption/encryption.

G: SBC-A receives the answer SDP that has the cookie sent by SBC-B.

H: SBC-A decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-A adds the SDES crypto attribute retrieved from the cookie to the answer SDP. SBC-A does not install any SA or media policy so that SRTP packets from/to UA-A can pass through SBC-A without any decryption/encryption.

Music on Hold

After a call is established, one of the endpoints can put the other endpoint on hold. An application server might intercept the re-INVITE from one endpoint (for putting the other on hold) and implement MoH as follows.



A: Endpoint UA-A sends an offer SDP for hold to SBC-A.

B: SBC-A forwards offer SDP without any cookie since there will be no media going from/to UA-A.

C: SBC-B receives an offer SDP that is addressed to the MS without any cookie.

D: Because there is no cookie in the ingress offer SDP, SD-B generates its own SDES crypto attributes for the egress offer SDP sent to UA-B.

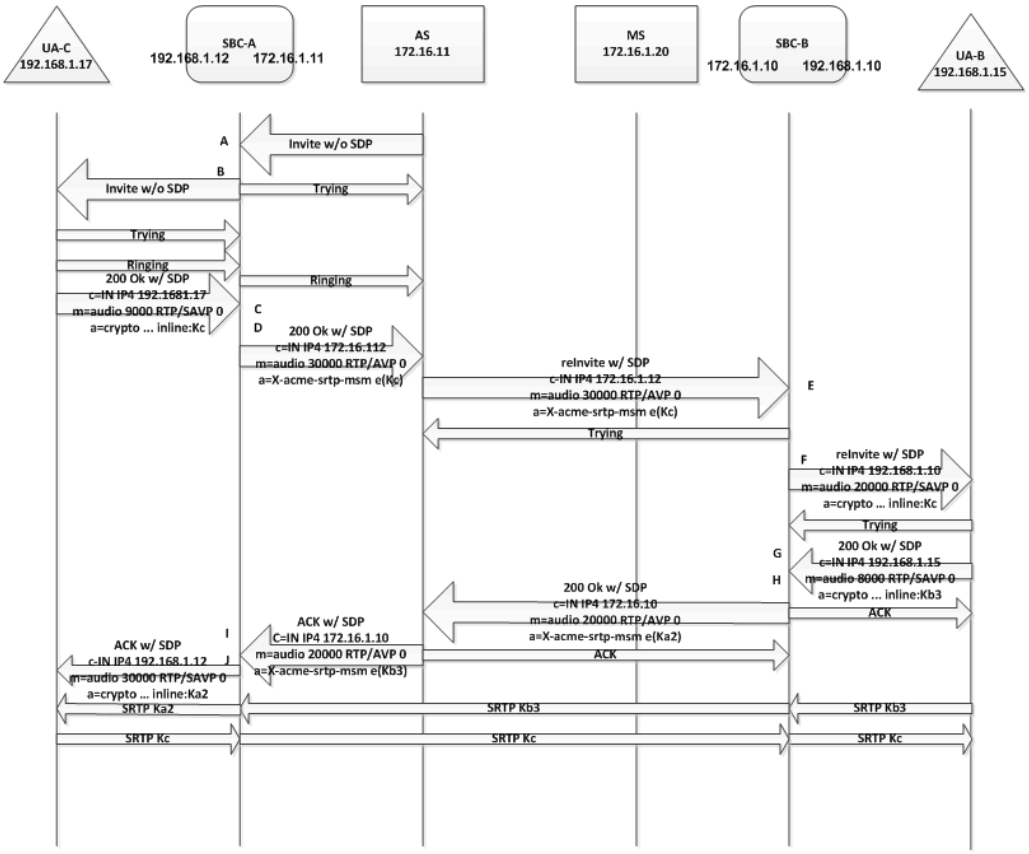
E: SBC-B receives am answer SDP from UA-B.

F: SBC-B sends its answer SDP without any cookie and encrypts SRTP packets going to UA-B. Note that there will be no SRTP packets going from UA-B to SBC-B.

As a result, MoH is heard by UA-B as it decrypts SRTP packets encrypted by SBC-B. When UA-A resumes, the steps to establish media between UA-A and UA-B will be the same as the steps used for call setup.

Once the media is reestablished between UA-A and UA-B media travelling between the two UAs will not be decrypted by either SBC.

Call Transfer



A call transfer can also happen after a call is established. For example, endpoint UA-B can transfer UA-A to another endpoint, UA-C. UA-B can make a blind transfer or an attended transfer. The following diagram describes a blind call transfer with SRTP Pass-through enabled. Note it does not show the SIP message exchange between UA-B and the Application Server to facilitate the call transfer (i.e. the INVITE from UA-B to put the call on hold and the REFER/ NOTIFY message exchange between UA-B and the Application Server). After UA-A is put on hold and the transfer target (that is, UA-C) is received from UA-B, the Application Server attempts to establish a call to UA-C. After the call to UA-C is established, the Application Server takes UA-A off hold and connects its media session to that of UA-C.

A: SBC-B receives an INVITE (from the Application Server to invite UA-C) without an offer SDP.

B: SBC-B sends an INVITE without an offer SDP to UA-C.

C: SBC-B receives a 200 OK response with an offer SDP that has a SDES crypto attribute on the SRTP media line from UA-C.

D: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SBC-B adds a cookie to the egress offer SDP that is sent to the core realm.

E: SBC-A receives a re-INVITE to UA-A with the offer SDP that has the cookie sent by SBC-B.

F: Since Multi-system Selective SRTP Pass-through is enabled within the ingress realm, SBC-A adds the SDES crypto attribute retrieved from the cookie to the offer SDP sent to UA-A.

G: SBC-A receives an answer SDP that has a SDES crypto attribute on the SRTP media line from UA-A.

H: SBC-A checks if it has a cookie in the offer SDP and adds the cookie to the answer SDP that is sent to the core realm. The cookie contains an encryption of the SDES crypto attribute received in the answer SDP from UA-B. SBC-A stops the encryption of any SRTP packets going to UA-B (set up for MoH) so that SRTP packets from/to UA-B can now pass through SBC-A without any decryption/encryption.

I: SBC-B receives the answer SDP that has the cookie sent by SBC-A.

H: SBC-B decrypts the cookie using the shared secret to retrieve the SDES crypto attribute. SBC-B adds the SDES crypto attribute retrieved from the cookie to the answer SDP.

After the call to UA-C is established and the call to UA-A is resumed (with the resulted media going between UA-A and UA-C) the Application Server disconnects the call with UA-A. Note that steps C-J are essentially the same steps as steps A-H in the Call Setup example. The difference is that the offer SDP from C comes to SD-B in a 200 OK response and the answer SDP sent by SBC-B to C is in the ACK.

Early Media

Different application servers may implement early media in different ways. In general, the SBC supports early media in the following way.

If the SBC receives an SDP without any cookie in a provisional response, the SBC generates its own SRTP crypto context and exchanges it with the caller via the SDP included in the provisional response. The SBC continues to decrypt and encrypt early media packets going to and from the caller. The SBC stops decrypting and encrypting only if it receives a final response with an answer SDP that signals that SRTP Pass-through should be enabled.

Multi-system Selective SRTP Pass-through with Media Release

When SRTP Pass-through is allowed, the hair-pinned media can also be released to the network if media release is configured — that is, if realm-config parameters **msm-release** is **enabled**, **mm-in-realm** is **disabled**, or **mm-in-network** is **disabled**. If SRTP Pass-through is not allowed, media release will not be allowed either.

Multi-system Selective SRTP Pass-through Configuration

Use the following procedure to enable Multi-system Selective SRTP Pass-through within a specific realm.

1. Use the following command sequence to move to **realm-config** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
```

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **srtp-msm-passthrough** parameter to enable Multi-system Selective SRTP Pass-through within a specific realm.

   By default, pass-through support is disabled.

   ```
   ORACLE(realm-config)# srtp-msm-passthrough enabled
   ORACLE(realm-config)#
   ```

3. Use **done**, **exit**, and **verify-config** to complete enabling Multi-system Selective SRTP Pass-through within the current realm.

   verify-config checks that the **srtp-msm-password** parameter has been configured, and outputs an error if it has not been configured. **verify-config** also checks other configuration settings that conflict with Multi-system SRTP Pass-through operation. Among these possible mis-configurations are the following.

   rfc2833-mode set to preferred on a SIP interface within a realm that has srtp-msm-passthrough enabled

   rfc2833-mode set to preferred and app-protocol set to SIP on a session-agent within a realm that has srtp-msm-passthrough enabled.

4. If required, repeat Steps 1 through 3 to enable Multi-system Selective SRTP Pass-through on additional realms.

   Use the following procedure to specify values needed to support the exchange of SDES keying information.

5. Use the following command sequence to move to **security** Configuration Mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)#
   ```

6. Use the **srtp-msm-attr-name** parameter to specify the name of the media attribute used to convey SDES keying information within a SDP media description.

   A valid attribute name must consist of characters from the US-ASCII subset of ISO-10646/UTF-8 as specified in RFC 2327, *SDP: Session Description Protocol*. IANA-registered names should not be used. Values should begin with an X-1 prefix to prevent collision with registered values.

   In the absence of a specified attribute name, the SD provides a default value of X-acme-srtp-msm.

   ```
   ORACLE(security)# srtp-msm-attr-name X-key-material
   ORACLE(security)#
   ```

7. Use the **srtp-msm-password** parameter to provide the shared secret used to derive the key for encrypting SDES keying material that is placed in the media attribute of an SDP media description. Ingress keying material is encrypted using this shared secret before being forwarded to the network core. On egress, the encrypted keying material is decrypted with this same key.

   Allowable values are characters strings that contain a minimum of 8 and a maximum of 16 characters.

   ```
   ORACLE(security)# srtp-msm-password IsHeEleemosynary
   ORACLE(security)#
   ```

8. Use **done**, **exit**, and **verify-config** to complete necessary configuration.

verify-config checks that the **srtp-msm-password** parameter has been configured, and outputs an error if it has not been configured. **verify-config** also checks other configuration settings that conflict with Multi-system SRTP Pass-through operation. Among these possible mis-configurations are the following.

rfc2833-mode set to preferred on a SIP interface within a realm that has srtp-msm-passthrough enabled

rfc2833-mode set to preferred and app-protocol set to SIP on a session-agent within a realm that has srtp-msm-passthrough enabled.

Statistics

The number of media sessions set up with Multi-systems Selective SRTP Pass-through is tracked and included in the output of the **show mbcd statistics** command.

SRTP and Transcoding

Secure Real Time Transport Protocol (SRTP) allows secure media transmission. Transcoding is the ability to convert between media streams that are based upon different codecs. The Oracle USM supports IP-to-IP transcoding for SIP sessions and can connect two voice streams that use different coding algorithms with one another. Both SRTP and transcoding are available in the same call. This feature is not available on the Acme Packet 3820 or Acme Packet 4500 platforms.

As of this release of the software, SRTP and transcoding may be combined for the same call. This behavior is available by default and no extra configuration is required.

SRTP Re-keying

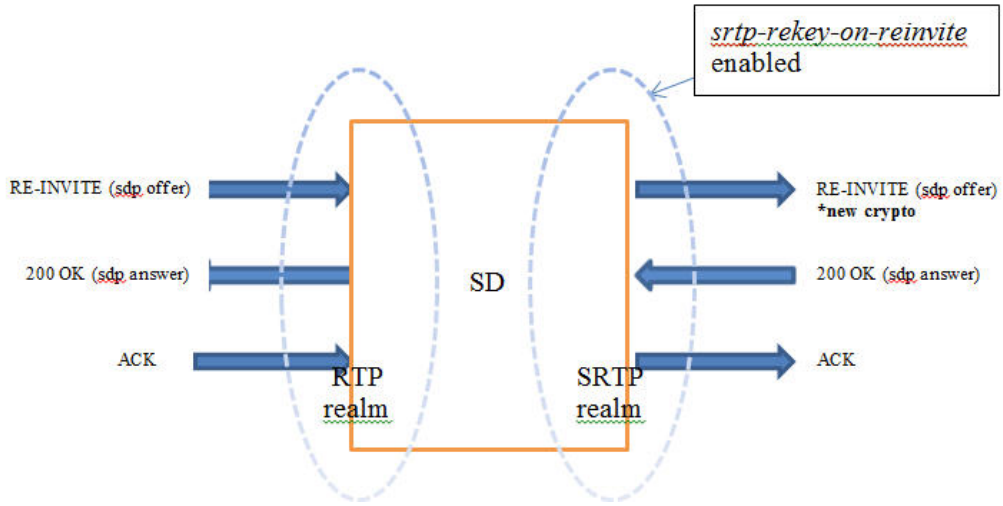
Initialization of SRTP re-keying is supported by the Oracle USM.

The Oracle USM can generate a new outbound crypto attribute in the SDP offer in a SIP re-INVITE when the **srtp-rekey-on-reinvite** parameter is set to **enabled**. The system generates the attribute regardless of the state of the flow, active or not.

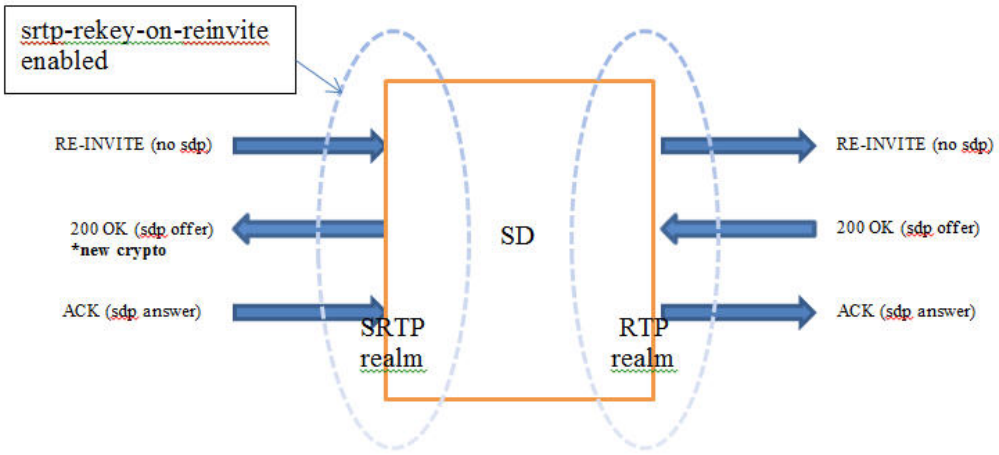
This capability is important for some clients that reside on the SRTP side in a single SRTP termination mode configuration. Any media changes that happen in the RTP side are hidden by the Oracle USM. This concealment may cause issues in some configurations, where media servers are involved. When the media changes from media server to called phone, the SRTP endpoint is not aware the media source changed because the SDP offer from the Oracle USM is the same as original invite. The result is that some devices drop packets because of Synchronization Source Identifier (SSRC) values mismatch, unexpected jumps in sequence number, sequence number reversions back to 1 triggering replay attack defense, and so forth. In certain environment is has been found that re-keying on every re-invite eliminates all these issues especially in customer setups that use Microsoft Lync products.

The processing of standard RE-INVITES (those containing an SDP offer) and offerless RE-INVITES is shown below.

With SDP:



No SDP:



If the re-invite message is a refresh and **srtp-rekey-on-reinvite** is enabled, the outbound crypto will change but the SDP version will not be incremented on the outgoing invite. If this scenario causes incompatibility issues with customer equipment then add the unique-sdp-id option to media-manager->option configuration so the Oracle USM increments the SDP version in the outgoing invite.

SRTP Re-keying Configuration

Configure **srtp-rekey-on-reinvite** to enable the negotiation and generation of new SRTP keys upon the receipt of a SIP RE-INVITE message that contains SDP.

Confirm that an **sdes-profile** exists.

In the following procedure, change the default state to enabled.

1. Access the **sdes-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# media-security
ORACLE(media-security)# sdes-profile
ORACLE(sdes-profile)#
```

2. Type **select** to choose and configure an existing object.

```
ORACLE(sdes-profile)# select
<name>:
1:  name=sdesprofile01

selection: 1
ORACLE(sdes-profile)#
```

3. **srtp-rekey-on-reinvite**—Set this parameter to **enabled** for re-keying upon the receipt of an SIP reINVITE that contains SDP.
4. Type **done** to save your configuration.

IPSec Support

The Oracle USM offers IPSec for securing signaling, media, and management traffic at the network layer.

Supported Protocols

The Oracle USM's IPSec implementation supports all required tools for securing Internet communication via the IPSec protocol suite. The following paragraphs list and explain the protocols within the IPSec suite that the Oracle USM supports. This chapter does not explain how to design and choose the best protocols for your application.

AH vs. ESP

The Oracle USM supports the two encapsulations that IPSec uses to secure packet flows. Authentication Header (AH) is used to authenticate and validate IP packets. Authentication means that the packet was sent by the source who is assumed to have sent it.

> **Note:**
>
> AH is incompatible with NAT. Validation means that the recipient is assured that the packet has arrived containing the original, unaltered data as sent.

ESP (Encapsulating Security Payload) provides AH's authentication and validations and extends the feature set by ensuring that the IP packet's contents remain confidential as they travel across the network. Using an encryption algorithm that both peers agree upon, ESP encrypts a full IP packet so that if intercepted, an unauthorized party cannot read the IPSec packet's contents.

Tunnel Mode vs. Transport Mode

In addition to its security encapsulations, the IPSec suite supports two modes: tunnel mode and transport mode. Tunnel mode is used most often for connections between gateways, or between a host and a gateway. Tunnel mode creates a VPN-like path between the two gateways and encapsulates the entire original IP packet. Transport mode is used to protect end-to-end communications between two hosts providing a secured IP connection and encrypts just the original payload.

> **Note:**
>
> Traffic sent through the inner IPSec tunnel must be on the same VLAN-slot-port network-interface combination as where the outer tunnel is configured. This is because IPSec tunnel mode does not carry any L2 information for the inner packet. Once the SBC decrypts (de-tunnel) the received packet, it uses the L2 header from the original packet for the lookup. Therefore, if the SBC uses different vlan/slot/port for the inner network, lookups will fail.

Cryptographic Algorithms

IPSec works by using a symmetric key for validation and encryption. Symmetric key algorithms use the same shared secret key for encoding and decoding data on both sides of the IPSec flow. The Oracle USM's IPSec feature supports the following encryption algorithms:

- DES
- 3DES
- AES128CBC
- AES256CBC
- AES128CTR
- AES256CTR

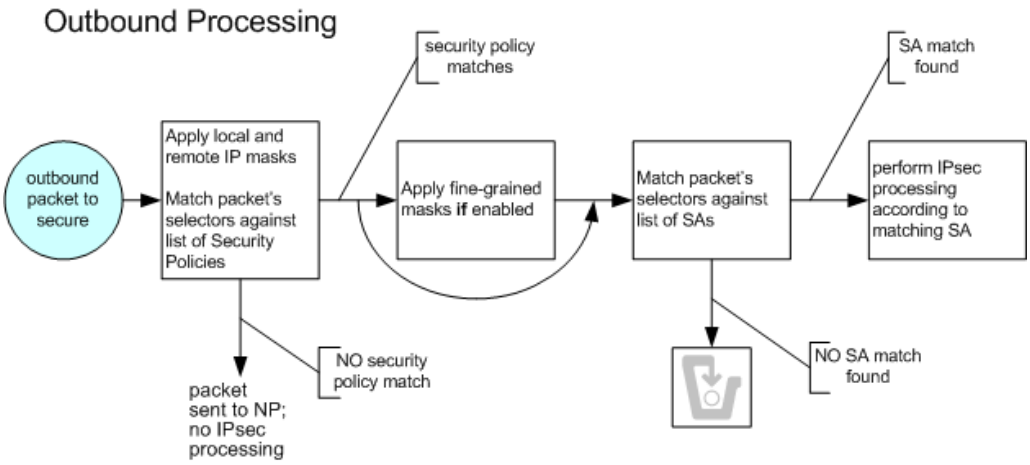
The Oracle USM can quickly generate keys for all of the above mentioned algorithms from the CLI. It can additionally support HMAC-SHA1 or HMAC-MD5 keyed-hash message authentication codes. Only manual keying is currently supported for both hash authentication and data encryption. Therefore, all keys must be provisioned on the Oracle USM by hand.

IPSec Implementation

The Oracle USM uses separate logic for processing IPSec packets based on whether the traffic is inbound or outbound. The configuration is divided into two pieces, the security policy and the security association (SA). Both the SA and security policies have a directional attribute which indicates if they can be used and/or reused for inbound and outbound traffic.

Outbound Packet Processing

The following diagrams show the steps the Oracle USM follows when processing outbound IPSec traffic. Details of each step are described in the following sections.



Security Policy

The Oracle USM first performs a policy lookup on outbound traffic to test if it should be subjected to IPSec rules. A security policy, local policy applicable for IPSec functionality, defines the matching criteria for outgoing network traffic to secure. It is configured on a network interface basis.

Configuring a security policy is similar to a local policy, with additional IPSec-specific parameters. Unlike a local policy, used for routing, a security policy is used for packet treatment. As with a local policy, a set of selector values is matched against the outbound flow's following characteristics:

- VLAN
- Source IP address (plus mask)
- Source IP port
- Destination IP address (plus mask)
- Destination IP port
- Transport Protocol

Each of these selection criteria can be defined by a wildcard except for the VLAN ID, which can be ignored. This flexibility aids in creating selection criteria that ranges from highly restrictive to completely permissive. In addition to the main traffic matching criteria, a priority parameter is used to prioritize the order that configured security policies are checked against. The #0 policy is checked first, #1 policy is checked next, continuing to the lowest prioritized policy being checked last.

Once the outbound traffic matches a policy, the Oracle USM proceeds to the next step of outbound IPSec processing. If no matching security policy is found, the default pass-through policy allows the packet to be forwarded to the network without any security processing.

Fine-grained policy Selection

After a positive match between outbound traffic and the configured selectors in the security policy, the Oracle USM can perform a calculation between a set of fine-grained packet selectors and the outbound packet. The fine-grained policy masking criteria are:

- Source IP subnet mask

- Destination IP subnet mask
- VLAN mask

By default, the fine-grained security policy is set to match and pass all traffic untouched to the security association (SA) portion of IPSec processing.

Fine-grained policy selection works by performing a logical AND between outbound traffic's fine-grained selectors and the traffic's corresponding attributes. The result is then used to find the matching SA. Applying a fine-grained mask has the effect of forcing a contiguous block of IP addresses and/or ports to appear as one address and or port. During the next step of IPSec processing, when an SA is chosen, the Oracle USM in effect uses one SA lookup for a series of addresses. Without fine-grained policy selection, unique SAs must always be configured for outbound packets with unique security policy selectors.

Security Associations

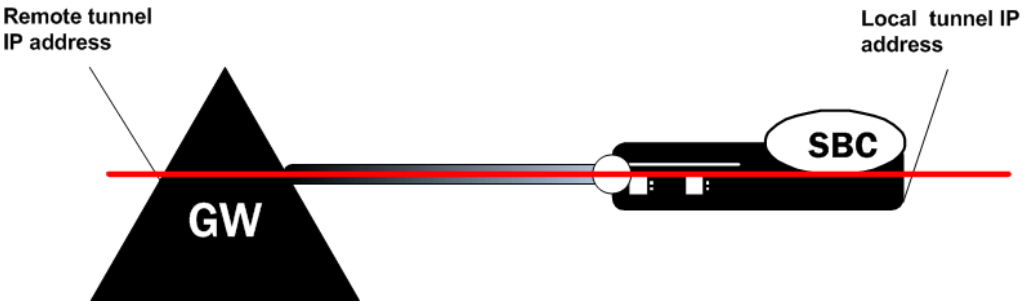
After the Oracle USM determines that outgoing traffic is subject to IPSec processing, and optionally applies fine-grained masking, an SA lookup is performed on the traffic. An SA is the set of rules that define the association between two endpoints or entities that create the secured communication. To choose an SA, the Oracle USM searches for a match against the outgoing traffic's SA selectors. SA selectors are as follows:

- VLAN
- Source IP address
- Source IP port
- Destination IP address
- Destination IP port
- Transport Protocol

If there is a match, the Oracle USM secures the flow according to security parameters defined in the SA that the Oracle USM chooses. The packet is then forwarded out of the Oracle USM. If no match is found, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG.

Secure Connection Details

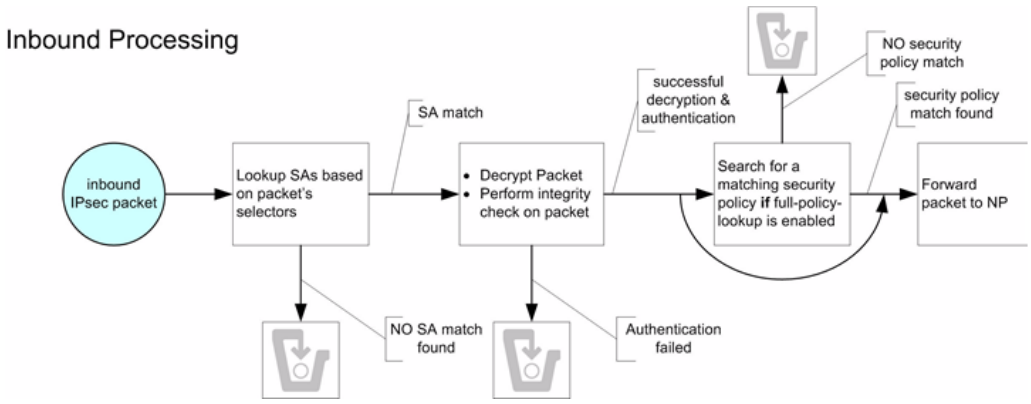
Several parameters define an IPSec connection between the Oracle USM and a peer. When planning an IPSec deployment, the primary architectural decisions are which IPSec protocol and mode to use. The two choices for IPSec protocol are ESP or AH, and the two choices for IPSec mode are either tunnel or transport. IPSec protocol and mode are both required for an SA configuration. When creating an IPSec tunnel (tunnel mode), the SA must also define the two outside IP addresses of the tunnel.



The authentication algorithm and the authentication key must always be configured. The Oracle USM supports hmac-md5 or hmac-sha1 authentication algorithms. Because only manual keying is supported, the key must be entered by hand. When encryption is required, the encryption algorithm and the encryption key must be configured. The Oracle USM supports des, 3des, aes-128-cbc, aes-256-cbc, aes-128-ctr, and aes-256-ctr encryption algorithms. When using the two encryption protocols that operate in AES counter mode (RFC 3686), an additional nonce value is required. In addition, the security parameter index (SPI) must be configured for each SA. All SPI values mst be unique as well, across all SAs.

Inbound Packet Processing

The following diagram shows the steps the system follows when processing inbound IPSec traffic. Details of each step are described in the following sections.



IP Header Inspection

Processing inbound IPSec packets begins by the Oracle USM inspecting an inbound IP packet's headers. If the packet is identified as IPSec traffic, as determined by the presence of an AH or ESP header, an SA policy lookup is performed. If the traffic is identified as non-IPSec traffic, as determined by the lack of an IPSec-type (AH or ESP) header, it still is subject to a policy lookup. However, due to the default allow policy, the packet is forwarded directly to the NP, without any security processing.

SA Matching

The Oracle USM proceeds to match the inbound IPSec traffic's selectors against configured SAs. Inbound selector masking is performed where noted. These selectors are:

- VLAN (plus mask)
- Source IP address (plus mask)
- Source IP port
- Destination IP address (plus mask)
- Destination IP port
- Transport Protocol

- SPI

If no matching SA is found, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG. When the Oracle USM finds a matching SA, the packet is authenticated and decrypted according to the configuration and sent to the Oracle USM's NP for continued processing.

Inbound Full Policy Lookup

Inbound traffic can optionally be subjected to a full policy lookup, after decryption and authentication. A full policy lookup checks if a security policy exists for this inbound traffic before the Oracle USM sends the decrypted packet to the NP. If no matching security policy is found, even after a successful SA match, the packets are discarded, and optionally dumped to secured.log if the log-level is set to DEBUG.

Full policy lookups are useful for traffic filtering. If you wish to drop traffic not sent to a specific port (e.g. drop any traffic not sent to port 5060), a security policy with specific remote-port-match parameter would be used to define what is valid (i.e., not dropped).

HA Considerations

Anti-replay mechanisms, running on IPSec peers, can cause instability with the Oracle USMs configured in an HA pair. The anti-replay mechanism ensures that traffic with inconsistent (non-incrementing) sequence numbers is labeled as insecure, assuming it could be part of a replay attack. Under normal circumstances, this signature causes the remote endpoint to drop IPSec traffic.

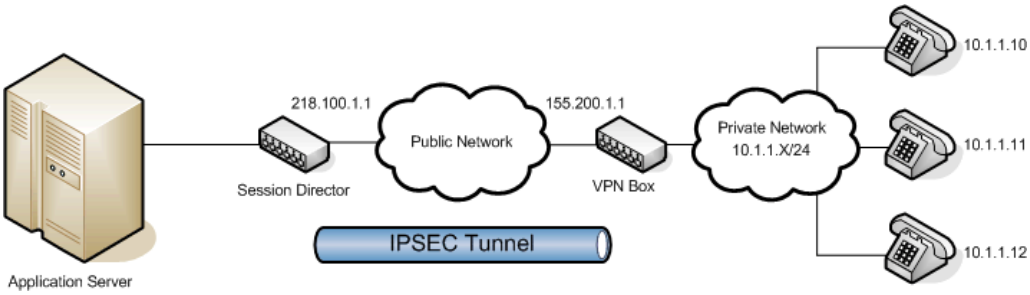
When a failover occurs between HA peers, the newly-active system starts sending traffic with the IPSec sequence number starting at 0. A remote system's anti-replay mechanism observes this and labels the traffic as insecure. It is therefore recommended that anti-replay protection not be used with Oracle USMs in an HA configuration. This situation does not create any problems as long as IPSec peers are not configured to use anti-replay mechanisms.

Packet Size Considerations

The security processor supports receipt of jumbo frames up to 9K (9022 bytes with VLANs, 9018 without). Under normal operation the default outgoing maximum packet size of 1500 bytes is used. This packet size includes the IPSec headers, which will result in less space for packet data (SIP signaling, RTP, etc...).

IPSec Application Example

In this example, the Oracle USM terminates an IPSec tunnel. The remote side of the tunnels is a dedicated VPN appliance in the public Internet. Behind that VPN appliance are three non-IPSec VoIP phones. In this scenario, the VPN box maintains the IPSec tunnel through which the phones communicate with the Oracle USM.



Without the fine-grained option (or alternatively IKE), an SA entry would need to be configured for each of the three phones, communicating over the IPSec tunnel (resulting in 3 tunnels).

This does not scale for manual-keying with a large number of endpoints. Using the fine-grained configuration as well as the inbound SA mask allows any number of endpoints on the 10.1.1.X network to use a single security association (a coarse-grain configuration). The configuration in this example follows:

A packet sent from the Oracle USM to any of the phones will match the policy pol1. The remote-ip-mask parameter of the fine-grained configuration will then be masked against the remote-ip, resulting in a SA selector value of 10.1.1.0. This matches security-association sa1, and the packet will be secured and sent over the tunnel. The tunnel-mode addresses in the security-association represent the external, public addresses of the termination points for the IPSec tunnel.

Packets returning from the 10.1.1.0 side of the IPSec tunnel will first match the tunnel-mode local-ip-addr of 218.100.1.1. The packets will then be decrypted using the SA parameters, and the tunneled packet will be checked against the remote-ip-addr field of the SA.

If the fine-grained mask had not been used, three discrete SAs would have to be configured: one for each of the three phones.

```
ORACLE(manual)# show
        manual
                name                            assoc1
                spi                             1516
                network-interface               lefty:0
                local-ip-addr                   100.20.50.7
                remote-ip-addr                  100.25.56.10
                local-port                      60035
                remote-port                     26555
                trans-protocol                  ALL
                ipsec-protocol                  esp
                direction                       both
                ipsec-mode                      tunnel
                auth-algo                       hmac-md5
                encr-algo                       des
                auth-key
                encr-key
                aes-ctr-nonce                   0
                tunnel-mode
                        local-ip-addr                   100.20.55.1
                        remote-ip-addr                  101.22.54.3
                last-modified-date              2007-04-30 16:04:46
```

IPSec Configuration

The following example explains how to configure IPSec on your Oracle USM.

> **Note:**
>
> If you change the phy-interface slot and port associated with any SAs or SPDs, the Oracle USM must be rebooted for the changes to take effect.

Configuring an IPSec Security Policy

To configure an IPSec security policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter to access the security path of the configuration menu.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ipsec** and press Enter.

   ```
   ORACLE(security)# ipsec
   ORACLE(ipsec)#
   ```

4. Type **security-policy** and press Enter. The prompt changes to let you know that you can begin configuration.

   ```
   ORACLE(ipsec)# security-policy
   ORACLE(security-policy)#
   ```

5. **name**—Enter a name for this security policy. This parameter is required and has no default.
6. **network-interface**—Enter the network interface and VLAN where this security policy applies in the form: interface-name:VLAN
7. **priority**—Enter the priority number of this security policy. The default value is zero (**0**). The valid range is:
   - Minimum—0
   - Maximum—254
8. **action**—Enter the action the Oracle USM should take when this policy matches outbound IPSec traffic. The default value is **ipsec**. The valid values are:
   - **ipsec**—Continue processing as IPSec traffic
   - **allow**—Forward the traffic without any security processing
   - **discard**—Discard the traffic
9. **direction**—Enter the direction of traffic this security policy can apply to. The default value is **both**. The valid values are:
   - **in**—This security policy is valid for inbound traffic
   - **out**—This security policy is valid for outbound traffic

- **both**—This security policy is valid for inbound and outbound traffic

To define the criteria for matching traffic selectors for this security policy:

10. **local-ip-addr-match**—Enter the source IP address to match. The default value is **0.0.0.0**.
11. **remote-ip-addr-match**—Enter the destination IP address to match. The default value is **0.0.0.0.**
12. **local-port-match**—Enter the source port to match. A value of 0 disables this selector. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—65535
13. **remote-port-match**—Enter the destination port to match. A value of 0 disables this selector. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—65535
14. **trans-protocol-match**—Enter the transport protocol to match. The default value is **all**. The valid values are:
    - **UDP | TCP | ICMP | ALL**
15. **local-ip-mask**—Enter the source IP address mask, in dotted-decimal notation. The default value is **255.255.255.255**.
16. **remote-ip-mask**—Enter the remote IP address mask, in dotted-decimal notation. The default value is **255.255.255.255.**
17. Save your work using the ACLI **done** command.

Defining Outbound Fine-Grained SA Matching Criteria

To define outbound fine-grained SA matching criteria:

1. From within the security policy configuration, type **outbound-sa-fine-grained-mask** and press Enter. The prompt changes to let you know that you can begin configuration.
2. **local-ip-mask**—Enter the fine-grained source IP address mask to apply to outbound IP packets for SA matching. Valid values are in dotted-decimal notation. The default mask matches for all traffic.
3. **remote-ip-mask**—Enter the fine-grained destination IP address mask to apply to outbound IP packets for SA matching. Valid values are in dotted-decimal notation. The default mask matches for all traffic.
4. **local-port-mask**—Enter the local port mask for this security policy. The default value for this parameter is **0**. The valid range is:
    - Minimum—0
    - Maximum—65535
5. **remote-port-mask**—Enter the remote port mask for this security policy. The default value for this parameter is **0**. The valid range is:
    - Minimum—0
    - Maximum—65535
6. **trans-protocol-mask**—Enter the transport protocol mask for this security policy. The default value for this parameter is **0**. The valid range is:

- Minimum—0
- Maximum—255

7. **vlan-mask**—Enter the fine-grained VLAN mask to apply to outbound IP packets for SA matching. The default is **0x000 (disabled).** The valid range is:
   - 0x000 - 0xFFF
8. Save your work using the ACLI **done** command.

Configuring an IPSec SA

To configure an IPSec SA:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **security** and press Enter to access the security path of the configuration menu.

   ```
   ORACLE(configure)# security
   ORACLE(security)#
   ```

3. Type **ipsec** and press Enter.

   ```
   ORACLE(security)# ipsec
   ORACLE(ipsec)#
   ```

4. Type **security-association** and press Enter.

   ```
   ORACLE(ipsec)# security-association
   ORACLE(security-association)#
   ```

5. Type **manual** and press Enter. The prompt changes to let you know that you can begin configuration.

   ```
   ORACLE(security-association)# manual
   ORACLE(manual)#
   ```

6. **name**—Enter a name for this security policy.
7. **network-interface**—Enter the network interface and VLAN where this security policy applies in the form: interface_name:VLAN
8. **direction**—Enter the direction of traffic this security policy can apply to. The default value is **both**. Valid values are:
   - in | out | both
9. Save your work using the ACLI **done** command.

Defining Criteria for Matching Traffic Selectors per SA

To define the criteria for matching traffic selectors for this SA:

1. From within the **manual** portion of the security association configuration, you need to set the parameters described in this process.

   ```
   ORACLE(security-association)# manual
   ORACLE(manual)#
   ```

2. **local-ip-addr**—Enter the source IP address to match.
3. **remote-ip-addr**—Enter the destination IP address to match.

4. **local-port**—Enter the source port to match. A value of **0** disables this selector. The default value is **0**, disabling this parameter. The valid range is:
   - Minimum—0
   - Maximum—65535
5. **remote-port**—Enter the destination port to match. A value of **0** disables this selector. The default value is **0**, disabling this parameter. The valid range is:
   - Minimum—0
   - Maximum—65535
6. **trans-protocol**—Enter the transport protocol to match for traffic selectors for this SA. The default value is **ALL**. The valid values are:
   - **UDP | TCP | ICMP | ALL**
7. **ipsec-protocol**—Select the IPSec protocol to use for this SA configuration. The default value for this parameter is **esp**. Valid values are:
   - esp | ah
8. **spi**—Enter the security parameter index. The default value is **256**. The valid range is:
   - Minimum—256
   - Maximum—2302
9. **ipsec-mode**—Enter the IPSec mode of this SA. The default value is **transport**. The valid values are:
   - tunnel | transport
10. **auth-algo**—Enter the IPSec authentication algorithm for this SA. The default value is **null**. The valid values are:
    - hmac-md5 | hmac-sha1 | null
11. **auth-key**—Enter the authentication key for the previously chosen authentication algorithm for this SA.

> **Note:**
>
> The specified auth-key value will be encrypted in the configuration and will no longer be visible in clear-text.

12. **encr-algo**—Enter the IPSec encryption algorithm for this SA. The default value is **null**. The valid values are:
    - **des | 3des | aes-128-cbc | aes-256-cbc | aes-128-ctr | aes-256-ctr | null**
13. **encr-key**—Enter the encryption key for the previously chosen encryption algorithm for this SA.

> **Note:**
>
> The specified encr-key value will be encrypted in the configuration and will no longer be visible in clear-text.

14. **aes-ctr-nonce**—Enter the AES nonce if aes-128-ctr or aes-256-ctr were chosen as your encryption algorithm. The default value is **0**.

Defining Endpoints for IPSec Tunnel Mode

To define endpoints for IPSec tunnel mode:

1. From within the **manual** portion of the security association configuration, you need to set the parameters described in this process.

   ```
   ORACLE(security-association)# manual
   ORACLE(manual)#
   ```

2. **local-ip-addr**—Enter the local public IP address which terminates the IPSec tunnel.
3. **remote-ip-addr**—Enter the remote public IP address which terminates the IPSec tunnel.
4. Save your work using the ACLI **done** command.

Real-Time IPSec Process Control

The **notify secured** commands force the IPSec application to perform tasks in real-time, outside of the Oracle USM reloading and activating the running configuration. The **notify secured** usage is as follows:

```
notify secured [activateconfig | nolog | log | debug | nodebug]
```

The following arguments perform the listed tasks:

- **nolog**—Disables secured logging
- **log**—Enables secured logging
- **debug**—Sets secured log level to DEBUG
- **nodebug**—Sets secured log level to INFO

Key Generation

The **generate-key** command generates keys for the supported encryption or authentication algorithms supported by the Oracle USM's IPSec implementation. The generate-key commands generate random values which are not stored on the Oracle USM, but are only displayed on the screen. This command is a convenient function for users who would like to randomly generate encryption and authentication keys. The generate-key usage is as follows:

```
generate-key [hmac-md5 | hmac-sha1 | aes-128 | aes-256 | des | 3des]
```

IDS Reporting

The Oracle USM supports a wide range of intrusion detection and protection capabilities for vulnerability and attack profiles identified to date. The IDS reporting feature is useful for enterprise customers requirement to report on intrusions and suspicious behavior that it currently monitors.

IDS Licensing

This feature requires the IDS Reporting license.

> **Note:**
>
> The following capabilities and restrictions of the license:

- The following configuration parameters located in the **access control** and **media manager configuration** elements are only visible after installing the license:
  - trap-on-demote-to-deny
  - syslog-on-demote-to-deny
  - cac-failure-threshold
  - untrust-cac-failure-threshold
- Endpoint demotions based on admission control failures are only a valid option with the IDS License.
- The presence of the IDS license makes the apSysMgmtInetAddrWithReasonDOSTrap trap available and the apSysMgmtExpDOSTrap unavailable. WIthout an IDS license installed, only the apSysMgmtExpDOSTrap trap is available.
- The **Trust->Untrust** and **Untrust-Deny** counters in the SIP ACLs' statistics are visible regardless of the IDS license's presence.
- The **Demote Trust-Untrust** and **Demote Untrust-Deny** collect records in the SIP ACL HDR groups are visible regardless of the IDS license's presence.
- A GET operation can be preformed on the two MIB entries to view the global endpoint counter for Demotions from Trusted to Untrusted and from Untrusted to Deny regardless of the IDS license's presence

Basic Endpoint Demotion Behavior

Each session agent or endpoint is promoted or demoted among the trusted, untrusted, and denied queues depending on the **trust-level** parameter of the session agent or realm to which it belongs. Users can also configure access control rules to further classify signaling traffic so it can be promoted or demoted among trust queues as necessary.

An endpoint can be demoted in two cases:

1. Oracle USM receiving too many signaling packets within the configured time window (**maximum signal threshold** in **realm config** or **access control**)
2. Oracle USM receiving too many invalid signaling packets within the configured time window. (**invalid signal threshold** in **realm config** or **access control**)

Endpoint Demotion Reporting

The Oracle USM counts the number of endpoint or session agent promotions and demotions. Further, the Oracle USM counts when endpoints or session agents transition from trusted to untrusted and when endpoints transition from untrusted to denied queues. These counts are maintained for SIP signaling applications. They appear as the Trust->Untrust and Untrust->Deny rows in the **show sipd acls** command.

SNMP Reporting

These per-endpoint counters are available under APSYSMGMT-MIB -> acmepacketMgmt -> apSystemManagementModule -> apSysMgmtMIBObjects -> apSysMgmtMIBGeneralObjects.

MIB NAME	MIB OID	PURPOSE
apSysSipEndptDemTrustToUntrust	.1.3.6.1.4.1.9148.3.2.1.1.19	Global counter for SIP endpoint demotions from trusted to untrusted.
apSysSipEndptDemUntrustToDeny	.1.3.6.1.4.1.9148.3.2.1.1.20	Global counter for SIP endpoint demotions from untrusted to denied.

HDR Reporting

The SIP (sip-ACL-oper) HDR ACL status collection groups include the following two metrics:

- Demote Trust-Untrust (Global counter of endpoint demotion from trusted to untrusted queue)
- Demote Untrust-Deny (Global counter of endpoint demotion from untrusted to denied queue)

Endpoint Demotion SNMP Traps

An SNMP trap can be sent when the Oracle USM demotes an endpoint to the denied queue. This is set by enabling the **trap on demote to deny** parameter located in the **media manager config** configuration element.

When the IDS license is installed and the **trap on demote to deny** parameter is enabled, apSysMgmtInetAddrWithReasonDOSTrap trap is sent. This trap supersedes the apSysMgmtInetAddrDOSTrap trap.

When the IDS license is installed and the **trap on demote to deny** parameter is disabled the apSysMgmtInetAddrWithReasonDOSTrap trap is not sent from the Oracle USM, even when an endpoint is demoted to the denied queue.

This apSysMgmtInetAddrWithReasonDOSTrap contains the following data:

- apSysMgmtDOSInetAddressType—Blocked IP address family (IPv4 or IPv6)
- apSysMgmtDOSInetAddress—Blocked IP address
- apSysMgmtDOSRealmID—Blocked Realm ID
- apSysMgmtDOSFromURI—The FROM header of the message that caused the block (If available)
- apSysMgmtDOSReason—The reason for demoting the endpoint to the denied queue: This field can report the following three values:
  - Too many errors
  - Too many messages
  - Too many admission control failures

> **Note:**
>
> By default, this parameter is enabled for upgrade configurations, as the current behavior is to send a trap for every endpoint that is demoted to deny. However, for a new configuration created, the value to this configuration is disabled.

Trusted to Untrusted Reporting

Endpoints, however, transition to an intermediate state, untrusted, prior to being denied service. The Oracle USM provides an ACLI parameter, **trap-on-demote-to-untrusted**, that generates an SNMP trap when a previously trusted endpoint transitions to the untrusted state. Trap generation is disabled by default.

SNMP Reporting

Endpoint state transitions continue to be tracked by two counters available under APSYSMGMT-MIB -> acmepacketMgmt -> apSystemManagementModule -> apSysMgmtMIBObjects -> apSysMgmtMIBGeneralObjects.

MIB NAME	MIB OID	PURPOSE
apSysSipEndptDemTrustToUntrust	.1.3.6.1.4.1.9148.3.2.1.1.19	Global counter for SIP endpoint demotions from trusted to untrusted.
apSysSipEndptDemUntrustToDeny	.1.3.6.1.4.1.9148.3.2.1.1.20	Global counter for SIP endpoint demotions from untrusted to denied.

Endpoint Demotion Trusted-to-Untrusted SNMP Trap

The system can generate an SNMP trap when an endpoint transitions from the trusted to the untrusted state. The trap is structured as follows.

```
apSysMgmtInetAddrTrustedToUntrustedDOSTrap NOTIFICATION-TYPE
OBJECTS { apSysMgmtDOSInetAddressType,
apSysMgmtDOSInetAddress,
apSysMgmtDOSRealmID,
apSysMgmtDOSFromUri,
apSysMgmtDOSReason }
STATUS current
DESCRIPTION
"This trap is generated when an IP is placed on a untrusted list from trusted
list, and provides the ip address that has been demoted, the realm-id of that
IP, (if available) the URI portion of the SIP From header of the message that
caused the demotion."
::= { apSysMgmtDOSNotifications 5 }
```

The trap OID is 1.3.6.1.4.1.9148.3.2.8.0.5.

Endpoint Demotion Syslog Message

A Syslog message can be generated when an endpoint is demoted. Setting the **media manager config -> syslog-on-demote-to-deny** parameter to enabled writes and endpoint demotion warning to the syslog every time an endpoint is demoted to the denied queue. By default, this

configuration option is set to disabled. The syslog message has a WARNING Level and looks like this:

```
Jan 15 12:22:48 172.30.60.12 ACMESYSTEM sipd[1c6e0b90] WARNING SigAddr[access:
168.192.24.40:0=low:DENY] ttl=3632 guard=798 exp=30 Demoted to Black-List (Too
many admission control failures)
```

Event Log Notification Demotion from Trusted to Untrusted

You can enable your Oracle USM to provide event log notification (a syslog message) any time it demotes an endpoint from trusted to untrusted. The log message contains this data: IP address of the demoted endpoint, the endpoint's configured trust level, and the reason for demotion.

To use this feature, the intrusion detection system (IDS) Reporting license must be installed and your media manager configuration must be enabled to send the log messages. This feature is enabled with the **syslog-on-demote-to-untrusted** parameter in the media manager.

Endpoint Demotion Configuration

To configure the Oracle USM to send traps and/or write syslog messages on endpoint demotion:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **trap-on-demote-to-deny**—Set this parameter to enabled for the Oracle USM to send the apSysMgmtInetAddrWithReasonDOSTrap trap when applicable.
5. **syslog-on-demote-to-deny**—Set this parameter to enabled for the Oracle USM to write an endpoint demotion warning message to the syslog.
6. **syslog-on-demote-to-untrusted**—Change this parameter from **disabled** (default), to **enabled** so the Oracle USM will generate event notifications (syslog messages) when an endpoint becomes untrusted. For this capability to work, the IDS license must be installed on your system.
7. **trap-on-demote-to-untrusted**—Set this parameter to enabled for the Oracle USM to send the apSysMgmtInetAddrTrustedToUntrustedDOSTrap when the endpoint identified within the trap transitions from the trusted to untrusted state.
8. Save your work.

Endpoint Demotion due to CAC overage

The Oracle USM can demote endpoints from trusted to untrusted queues when CAC failures exceed a configured threshold. The Oracle USM can also demote endpoints from untrusted to denied queues when CAC failures exceed a another configured threshold.

The Oracle USM maintains CAC failures per-endpoint. The CAC failure counter is incremented upon certain admission control failures only if either one of **cac-failure-threshold** or **untrust-cac-fail-threshold** is non-zero.

The **cac failure threshold** parameter is available in the access control and realm configuration elements. Exceeding this parameter demotes an endpoint from the trusted queue to the untrusted queue. The untrust cac-failure-threshold parameter is available in the access control and realm configuration elements. Exceeding this parameter demotes an endpoint from the untrusted queue to the denied queue.

If both the **cac failure threshold** and **untrusted cac failure threshold** are configured to 0, then admission control failures are considered and counted as invalid signaling messages for determining if the **invalid-signal-threshold** parameter value has been exceeded.

CAC Attributes used for Endpoint Demotion

The Oracle USM determines CAC failures only by considering the calling endpoint's signaling messages traversing the calling realms' configuration. If an endpoint exceeds the following CAC thresholds, the Oracle USM will demote the endpoint when the CAC failure thresholds are enabled.

1. sip-interface user CAC sessions (realm-config > user-cac-sessions)
2. sip-interface user CAC bandwidth (realm-config > user-cac-bandwidth)
3. External policy server rejects a session

Authentication Failures used for Endpoint Demotion

If an endpoint fails to authenticate with the Oracle USM using SIP HTTP digest authentication OR endpoint fails authentication with an INVITE with authentication incase registration-caching is disabled, and receives back a 401 or 407 response from the registrar

When the Oracle USM receives a 401 or 407 message from the registrar in response to one of the following conditions, the endpoint attempting authentication is demoted.

- endpoint fails to authenticate with the Oracle USM using SIP HTTP digest authentication
- endpoint fails to authenticate with the Oracle USM using INVITE message when registration-caching is disabled

Endpoint Demotion Configuration on CAC Failures

To configure endpoint demotion on CAC failures:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **access-control** and press Enter.

   ```
   ORACLE(session-router)# access-control
   ORACLE(access-control)#
   ```

If you are adding this feature to an existing configuration, then you will need to select the configuration you want to edit.

4. **cac-failure-threshold**—Enter the number of CAC failures for any single endpoint that will demote it from the trusted queue to the untrusted queue.
5. **untrust-cac-failure-threshold**—Enter the number of CAC failures for any single endpoint that will demote it from the untrusted queue to the denied queue.
6. Save your work.

IDS Phase 2 (Advanced Reporting)

This feature supplements the IDS reporting and protection services. IDS Phase 2 provides enterprise users with additional tools to identify, monitor, and control suspicious, and possibly, malicious traffic patterns. IDS Phase 2 requires the IDS Advanced Reporting license.

License Requirements

IDS Phase 2 requires the IDS Advanced Reporting license, which is different from the IDS license introduced in the Oracle Communications S-C6.2.0 release. Access to new services described in this Technical Notice requires that the original IDS license be upgraded to the IDS Advanced Reporting license.

Rejected SIP Calls

IDS Phase 2 provides tools to monitor and record rejected SIP calls. A sudden or gradual increase in such calls can, but need not, indicate malicious intent.

IDS Phase 2 provides a global counter that increments with each SIP INVITE or REGISTER message that is rejected by the Acme Packet Oracle USM, and offers the option of generating a syslog message in response to call rejection.

Rejected Calls Counter

The rejected calls counter is a 32-bit global counter that records the total number of rejected SIP calls. Such calls have been rejected by the Oracle USM with the following response codes: 400, 403, 404, 405, 408, 413, 416, 417, 420, 423, 480, 481, 483, 484, 485, 488, 494, 500, 503, 505, and 604. These response codes may change with future software revisions.

The current value of the rejected calls counter is accessible via SNMP, Historical Data Recording (HDR), or the ACLI.

apSysMgmtGeneralObjects Table (1.3.6.1.4.1.9148.3.2.1.1)

Object Name	Object OID	Description
apSysSipTotalCallsRejected	1.3.6.1.4.1.9148.3.2.1.1.25	Global counter for SIP calls that are rejected by the SBC

The sip-error HDR collection group contains a new reporting field, Call Rejects, which contains the value of the global rejected calls counter.

The ACLI command **show sipd errors** displays the contents of the rejected calls counter.

```
ORACLE# show sipd errors
12:29:13-131
```

```
SIP Errors/Events             ---- Lifetime ----
                        Recent      Total  PerMax
SDP Offer Errors             0          0       0
SDP Answer Errors            0          0       0
Drop Media Errors            0          0       0
Transaction Errors           0          0       0
Application Errors           0          0       0
Media Exp Events             0          0       0
Early Media Exps             0          0       0
Exp Media Drops              0          0       0
Expired Sessions             0          0       0
Multiple OK Drops            0          0       0
Multiple OK Terms            0          0       0
Media Failure Drops          0          0       0
Non-ACK 2xx Drops            0          0       0
Invalid Requests             0          5       2
Invalid Responses            0          0       0
Invalid Messages             0          0       0
CAC Session Drop             0          0       0
Nsep User Exceeded           0          0       0
Nsep SA   Exceeded           0          0       0
CAC BW Drop                  0          0       0
Calls Rejected               0          0       0 <--
```

Syslog Reporting of Rejected Calls

Users can choose to send a syslog message in response to the rejection of a SIP call. In the default state, rejected calls are not reported to syslog.

Use the following ACLI command sequence to enable syslog reporting of rejected SIP calls.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# media-manager
ORACLE(media-manager-config)# syslog-on-call-reject enable
```

The syslog-on-call-reject attribute, which is disabled by default, enables the generation of a syslog message in response to the rejection of a SIP call.

Use **done**, **exit**, and **verify-config** to complete this configuration.

Syslog messages issued in response to call rejection contain the following call-related information.

- SIP status code indicating rejection cause
- SIP method name (INVITE or REGISTER)
- Reason for denial
- Realm of calling endpoint
- Applicable local response map
- Content of Reason header (if present)
- From URI of calling endpoint
- Target URI of called endpoint
- Source and Destination IP address and port
- Transport type

The following are sample syslog messages issued in response to call rejections.

Dec 8 06:05:42 172.30.70.119 deimos sipd[205bfee4] ERROR [IDS_LOG]INVITE from source 172.16.18.100:5060 to dest 172.16.101.13:5060[UDP] realm=net172; From=sipp <sip:sipp@172.16.18.100:5060>;tag=13890SIPpTag001; target=sip:service@172.16.101.13:5060 rejected!; status=483 (Too Many Hops)

Dec 10 15:09:28 172.30.70.119 deimos sipd[2065ace8] ERROR [IDS_LOG]INVITE from source 172.16.18.5:5060 to dest 172.16.101.13:5060[UDP] realm=net172; From=sipp <sip:sipp@172.16.18.5:5060>;tag=10015SIPpTag001; target=sip:service@172.16.101.13:5060 rejected!; status=488 (sdp-address-mismatch); error=sdp address mismatch

IDS syslog messages that report rejected calls and those that report endpoint demotions now contain a string IDS_LOG, to facilitate their identification as IDS-related messages. With IDS Phase 2, IDS messages reporting either endpoint demotions or call rejections can be sent to specific, previously-configured syslog servers.

In topologies that include multiple syslog servers, use the following procedure to enable delivery of IDS-related messages to one or more specific syslog servers.

1. Use the following command sequence to move to **syslog-config** Configuration Mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# system
   ORACLE(system)# system-config
   ORACLE(system-config)# syslog-servers
   ORACLE(syslog-config)#
   ```

2. From the existing pool of syslog servers select the server or servers that will receive syslog messages.
3. Ensure that all selected servers are configured with the same value for the **facility** attribute.

   Allowable values are integers within the range 0 through 23.

4. Use the following command sequence to move to **system-config** Configuration Mode.

   ```
   ORACLE(syslog-config)# done
   ORACLE(syslog-config)# exit
   ORACLE(system-config)#
   ```

5. Use the **ids-syslog-facility** attribute to enable message transfer to specific syslog servers.

   The default value, -1, disables selective message transfer. To enable transfer to a designated syslog server or servers, enter the facility value (an integer within the range 0 through 23) that you confirmed or set in Step 3.

   The following example enables the transfer of IDS syslog messages to all servers with a facility value of 16.

   ```
   ORACLE(system-config)# ids-syslog-facility 16
   ORACLE(system-config)#
   ```

6. Use **done**, **exit**, and **verify-config** to complete this configuration.

TCA Reporting of Denied Entries

Users can construct a Threshold Crossing Alarm (TCA) which issues minor, major, and critical system alarms when the count of denied entries exceeds pre-configured values. For each issued alarm, the TCA also transmits an SNMP trap that reports the alarm state to remote SNMP agents.

1. Use the following command sequence to move to **media-manager-config** Configuration Mode.

```
ORACLE# configure terminal
ORACLE(configure)# system
ORACLE(system)# system-config
ORACLE(system-config)# alarm-threshold
ORACLE(alarm-threshold)#
```

2. Use the **type** attribute to specify the TCS type (**deny-allocation** for denied entries TCAs), the **severity** attribute to specify the criticality of the alarm, and the **value** attribute to specify the alarm threshold.

   The following ACLI sequence defines three alarm thresholds (minor, major, and critical), although it is not necessary to configure all thresholds. Given the static deny allocation value of 32000 on an Acme Packet 4500 or 4500 and 16000 on an Acme Packet 3800, you can determine what the percentage value maps to.

```
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity minor
ORACLE(alarm-threshold)# value 80
ORACLE(alarm-threshold)# done
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity major
ORACLE(alarm-threshold)# value 90
ORACLE(alarm-threshold)# done
ORACLE(alarm-threshold)# type deny-allocation
ORACLE(alarm-threshold)# severity critical
ORACLE(alarm-threshold)# value 95
ORACLE(alarm-threshold)# done
```

3. Use **exit** and **verify-config** to complete this configuration.

   After issuing a system alarm and accompanying SNMP trap, the TCA continues to monitor the number of denied entries. If the number of denied entries rises to the next threshold value, a new, and more severe, system alarm/SNMP trap is generated. If the number of denied entries falls below the current threshold level, and remains there for a period of at least 10 seconds, a new, and less severe system alarm/SNMP trap is generated.

Syslog Reporting of Denied Entries

Syslog reporting of endpoint demotions was introduced as part of IDS Phase 1 in S-C6.2.0. With IDS Phase 2, such syslog messages contain the last SIP message from the endpoint that caused the transition to the denied state. If the included SIP message increases the length of the syslog beyond 1024 bytes, the SIP message is truncated so that the syslog is no larger than 1024 bytes.

CPU Load Limiting

The transmission of IDS-related system alarms and SNMP traps is disabled when the CPU utilization rate surpasses a configured threshold percentage. When the threshold is exceeded, a syslog message (MINOR level) announces the termination of IDS reporting. No additional syslog messages or SNMP traps are generated until the CPU utilization rate falls below the configured threshold. The resumption of IDS reporting is announced by another syslog message, also issued at the MINOR level.

By default the CPU utilization rate is 90%. This value can be changed by the following ACLI command sequence

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# options +load-limit="80"
ORACLE(sip-config)# done
```

Denied Endpoints

IDS Phase 2 provides a denied endpoint counter that includes SIP endpoints. The global counter value is available via SNMP or HDR.

The global counter value is available to SNMP under APSYSMGMT-MIB -> acmepacketMgmt -> apSystemManagementModule -> apSysMgmtMIBObjects -> apSysMgmtMIBGeneralObjects.

apSysMgmtGeneralObjects Table (1.3.6.1.4.1.9148.3.2.1.1)

Object Name	Object OID	Description
apSysCurrentEndptsDenied	1.3.6.1.4.1.9148.3.2.1.1.26	Global counter for current endpoints denied

The system HDR collection group contains a new reporting field, Current Deny Entries Allocated, which contains the value of the global endpoints denied counter.

Maintenance and Troubleshooting

show sipd acls

The **show sipd acls** command includes counters that track the number of endpoints demoted from trusted to untrusted and the number of endpoints demoted from untrusted to denied. For example:

```
ORACLE# show sipd acls
...
ACL Operations         ---- Lifetime ----
               Recent      Total  PerMax
...
Trust->Untrust       0          1       1
Untrust->Deny        0          1       1
```

17
Lawful Intercept

This section summarizes options for configuring the lawful intercept feature. It describes how the Oracle USM interoperates with mediation equipment from vendors who build lawful intercept equipment.

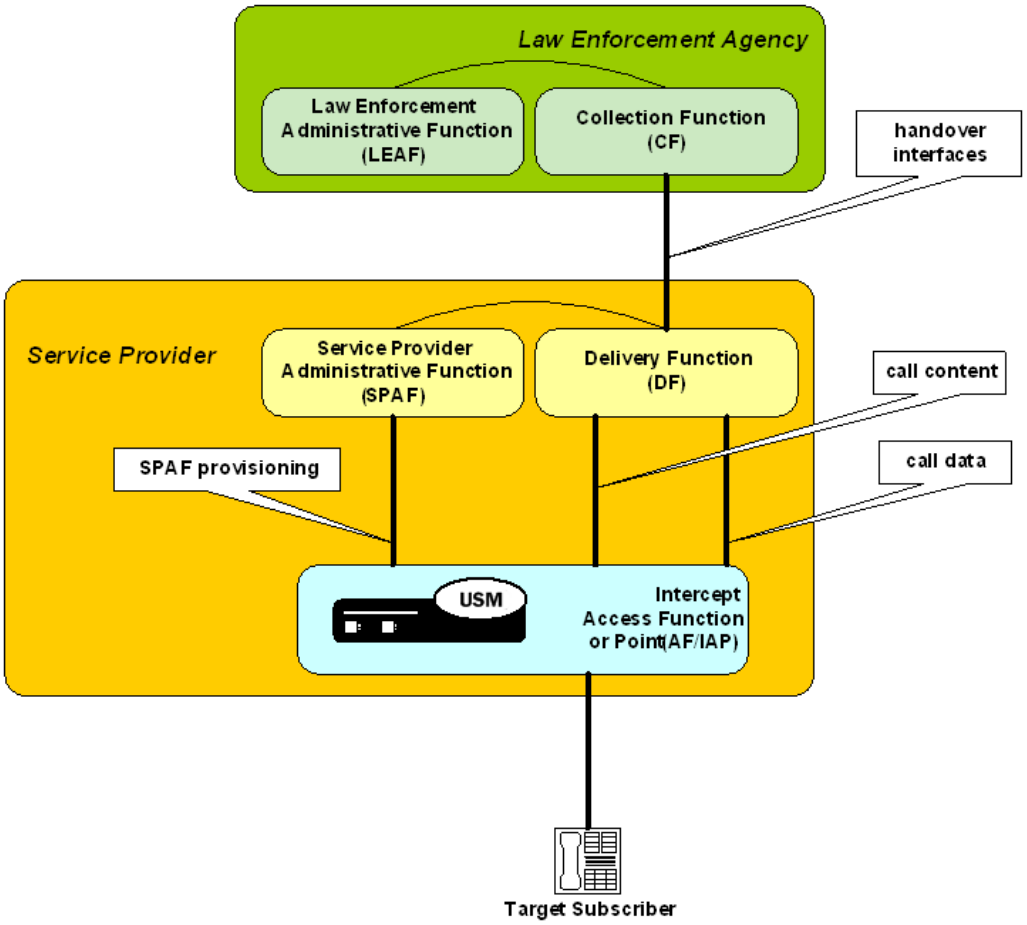
LI/CALEA consists of the interception of call content and/or the interception of call-identifying information. It requires that call information and media streams be sent to one or more law enforcement agencies in accordance with warrants for lawful interception.

You can configure your Oracle USM to support LI/CALEA functionality, enabling the Oracle USM to play a role in your Lawful Interception solution. Acting as an intercept access point (IAP), the Oracle USM can provide call data and can replicate media when signaling and media are anchored at the Oracle USM.

The Oracle USM supports LI/CALEA functionality that:

- Ensures unobtrusive intercept by hiding the network-based intercept of call information and content through topology hiding and media relay or NAT
- Intercepts and forwards call information and call content
- Interfaces with the mediation equipment [service provider administrative function (SPAF) and delivery function (DF)] for legal intercept

The following diagram provides one example of the Oracle USM deployed as part of a service provider's lawful intercept solution.



Recommendations

Calls may be lawfully intercepted by different devices in the service provider's network based on specific call flows, involvement of the device in the invoked service and where devices sit in the flow. Oracle recommends that you contact our Professional Services department to plan your use of the lawful intercept feature on your Oracle USM. Oracle Professional services can assist you with network/call flow analysis to determine which types of calls will involve the Oracle USM as an intercept access point and to recommend proper configuration.

Interoperability Using X1 X2 X3

This document describes how the Oracle USM supports X1, X2, and X3 interfaces for lawful interception of SIP calls. In this deployment, the Oracle USM acts as an interception point that receives provisioning information from an administrative function and, based on that information, provides call data and content. As with the other LI interoperability solutions that the Oracle USM supports, the X1, X2, and X3 interfaces ensure unobtrusive call intercept by hiding network-based intercept of call data and content. The Oracle USM supports intercept of call data only, or of call data and call content.

18

External Policy Servers

Diameter-based External Policy Servers

The Diameter base protocol (RFC 3588) is supported by the Oracle USM and is used for Resource and Admission Control Function (RACF) and Connectivity Location Function (CLF) applications. The existing licenses for COPS based CLF and RACF support Diameter and COPS.

Diameter Connection

The Oracle USM supports Diameter (RFC 3588) connections to a Diameter server over TCP. The base Diameter protocol runs on TCP port 3868. Diameter-based RACF and CLF are available from the media interfaces on the Oracle USM.

The Diameter connection is always initiated from the Oracle USM to the Diameter server. The Oracle USM begins the connection by sending a Capabilities-Exchange-Request (CER) to the server, which replies with Capabilities-Exchange-Answer (CEA) message.

HA Support

The Oracle USM's high availability (HA) capabilities support CAC. When one Oracle USM in an HA configuration goes out of service, the MAC addresses are reassigned to a healthy Oracle USM. IP addresses follow the MAC addresses to provide a seamless switchover between HA nodes.

After an HA failover, the Diameter connection on the primary Oracle USM is either gracefully torn down, or times out depending on behavior of the PDP. The backup Oracle USM attempts to create a new Diameter connection with the PDP.

FQDN Support

FQDNs are configured in the address parameter in the external policy server configuration element. These FQDNs must conform to RFC 1035. If the port parameter in external policy server configuration element is not zero, then it will be used to connect to the group of applicable external policy servers. If the port parameter is set to zero, then the port number returned in SRV RR from the DNS server will be used. When the Oracle USM queries a DNS server for an FQDN configured in the external policy server, it always appends the _diameter._tcp. to request only TCP based Diameter servers.

IPv6 Support

An external policy server configuration element with an application mode parameter set to Rx may be configured with an IPv6 address in the address parameter (in addition to an IPv4 address or FQDN).

Diameter Heartbeat

Device-Watchdog-Request (DWR) and Device-Watchdog-Answer (DWA)messages are used to detect transport failures at the application layer between the Oracle USM communicating with a policy server via Diameter. The request/answer message pair forms a heartbeat mechanism that can alert the requesting side if the answering side is not reachable.

The Oracle USM always responds to a DWR by replying with a DWA message. In addition, the Oracle USM can be configured to initiate DWR messages toward a policy server or other Diameter-based network device.

You configure the **watchdog ka timer** with a timeout value that determines the number of seconds a DWA is expected in response to the Oracle USM sending a DWR.

If the Oracle USM fails to receive a DWA response from a Policy Server within the configured interval, then the connection towards that Policy Server is considered failed and torn down. The Oracle USM attempts to recreate the TCP connection, followed by the recreating the Diameter connection by issuing a CEA toward the policy server.

Diameter Failures

During periods of application inactivity on the Diameter interface, Device-Watchdog-Request (DWR) and Answer (DWA) messages are exchanged between the client and server to provide an application-level heartbeat. DWRs may be sent toward the Oracle USM, which responds with a DWA message.

Prior to establishing a connection, the system tries to connect to a configured Diameter server using the default TCP retry interval of 10 seconds. This is also true if either side of the connection receives a TCP FIN or RST. The Oracle USM labels a connection as failed when it does not receive a DWR from the Diameter server within the guard timer period.

If an operational Diameter connection fails, the Oracle USM tries to re-open the TCP socket and Diameter connection to the Diameter server at 30 second intervals, and increases its retry interval to 5 minutes until a successful Diameter connection is made.

Application IDs and Modes

Diameter messages include an application ID to indicate the application and standards' body interface. The following table lists the different Application-IDs for the corresponding standards' and applications. Application IDs must be provisioned manually.

	Standards Reference Point			
	RACF			**CLF**
Reference Point/ Standards Body	Gq 3GPP R6 29.209	Rx 3GPP R7 29.214	Rq ETSI 283 026	e2 ETSI 283 035
Application-ID	16777222	16777229	16777222	16777231

You also set the application mode to specify the interface more precisely. Doing so avoids the potential for collision between interface types that can occur when you only configure the application identifier. By setting both the application mode and application identifier for the interface, you tell the Oracle USM the format for Diameter messages it sends and receives.

The following table describes the application mode settings.

Application Mode Type	Description
Rq	As the default mode for the interface, Rq is the Oracle USM's base RACF interface. Even when you leave the application mode set to none (default), the Oracle USM runs in Rq mode. The only exception to this rule is if you set the application identifier to 16777236 and leave the application mode set to none; in this instance, the interface runs in Rx mode.
Rx	The interface runs in Rx mode when you either: Set the application mode to Rx and the application identifier to 16777236 Leave the application mode set to none, and set the application identifier to 16777236
Gq	The interface runs in Gq mode when you set the application mode to Gq.
	Note: The application identifier 1677722 is no longer unique, but applies both to Rq and Gq interface modes.
e2	The interface runs in e2 mode, the base CLF interface, when you set the application mode to e2. Even if you leave the application mode set to none, the interface will run in e2 mode when the external policy server is configured as a CLF interface.
none	The interface runs in Rq mode when you do not configure an application identifier, or in Rx mode if you set the application identifier to 16777236.

IPv6 Support

The Oracle USM supports Diameter-based CLF and RACF external policy servers in both IPv4 and IPv6 networks. There are three areas of enhanced functionality where the Oracle USM's Diameter external policy server offerings have changed.

- AVP support of IPv6 addresses encoded in UTF-8 format
- Extra bandwidth allocation in policed flows to compensate for longer addresses
- Framed-IPv6-Prefix AVP (97) support

IPv6 Addresses in UTF-8 Format

When necessary, the Oracle USMchecks that IPv6 addresses are formatted correctly in UTF-8 when they are inserted into relevant AVPs. The applicable AVPs are

- Flow-Description AVP (507)
- Subscription-Id-Data AVP (444)
- Destination-Realm AVP (283) only if applicable

Framed-IPv6-Prefix AVP

The Oracle USM supports the Framed-IPv6-Prefix AVP (97) as defined in RFC4005.

The Diameter interface on the Oracle USM substitutes this AVP for the Framed-IP-Address AVP (8) in a Diameter message when carrying IPv6 addresses in the AVP.

The IP Address is encoded as Octet-String. Although the IPv6 address is a 128 bit number, it will not fall on a 4 byte boundary due to the formatting of this particular AVP. Additional whitespace will be added per RFC 3588 to the end of the octet-string to pad the ending of the AVP.

Bandwidth Allocation Compensation for IPv6

Transporting IPv6 packets requires extra bandwidth because of their larger packet size. This needs to be taken into account when allocating bandwidth and policing media. In order to reserve the necessary bandwidth for signaling messages, the **standard pkt rate** parameter has been added to the **media profile** configuration element.

The Oracle USM needs a baseline media packet size for bandwidth requests. It first checks if the SDP includes a **ptime** value. That value will be used as the baseline if present in the request. If this value is not included in the SDP, the **standard pkt rate** parameter is used as the baseline. The chosen value is then multiplied times 20 (the difference between an IPv4 and IPv6 packet).

This value is sent to the external policy server in the bandwidth request when sending media into an IPv6 realm. From that point it is used when allocating bandwidth or media policing.

Diameter: RACF

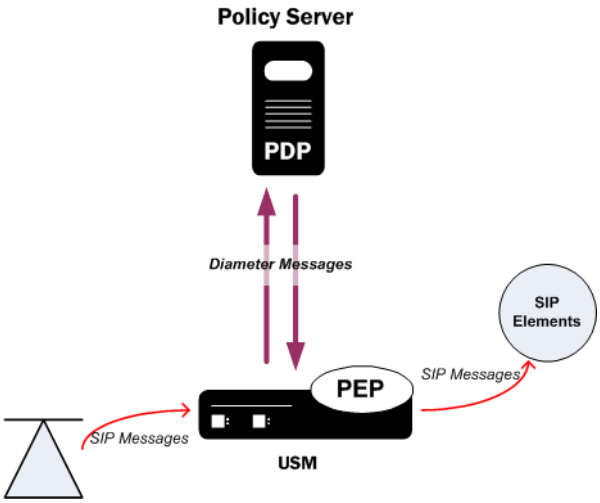
As the Oracle USM proxies and forwards calls, caller media information is known before the INVITE reaches the callee. The Oracle USM, acting as a P-CSCF, requests a specific amount of bandwidth for a call, and the RACF can reserve this amount of bandwidth for a call before the called phone rings. A call's required bandwidth can also be reserved by network devices along the path from the caller to callee if the QoS admission criteria is pushed from the RACF to other edge nodes (PEPs) such as routers, along this path to the callee.

Implementation Features

Bandwidth-based CAC is performed according to the following typical scenario. When the Oracle USM, known as the Policy Enforcement Point (PEP), receives a SIP INVITE, it sends a Diameter Authentication Authorization Request (AAR) message to the Policy Decision Point (PDP) or Resource and Admission Control Function (RACF). The Oracle USM does not forward the INVITE to its destination at this point.

The AAR message includes call identification information and the SDP-based bandwidth requirements for the call. The RACF responds with a Diameter Authentication Authorization Answer (AAA) message to either the install or remove the call. An install command directs the Oracle USM to forward the INVITE to the next SIP device. A remove command directs the Oracle USM send a SIP 503 Service Unavailable message sent back to the UA and reject the call.

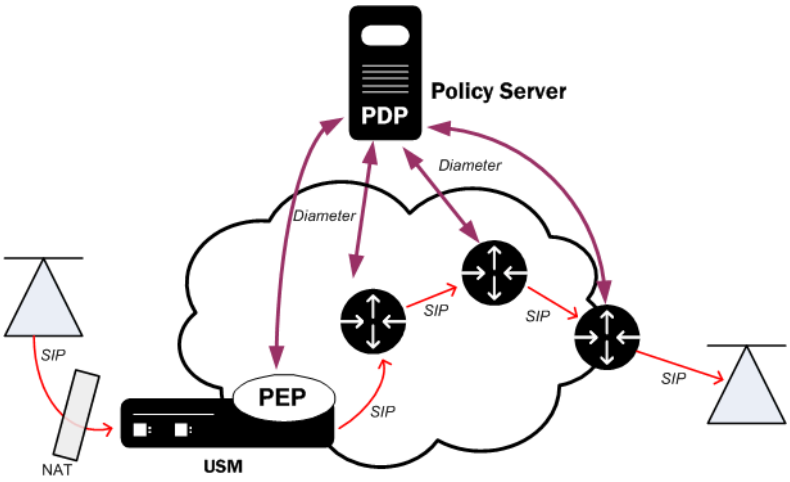
When the RACF is unreachable, incoming calls are rejected by default with a 503 message, as bandwidth can not be reserved. It is possible to configure the Oracle USM to allow all calls when the RACF is unreachable if this is the desired behavior.



The Oracle USM can be configured so that both sides of a call, based on realm, are subject to bandwidth enforcement. Each flow is treated as a unique call/event, because from a media and signaling perspective, they are distinct. As the Oracle USM functions as one side of a call, its IP address is inserted into the AAR message regardless of whether it is the calling or called party. This allows for the Diameter install or remove decision to be made before the Oracle USM receives the 200 OK response, and before ringing the far-end phone. Only one external policy server can be used within a single realm.

When a call ends, either with the expected SIP BYE or CANCEL methods, or due to other error conditions, the Oracle USM alerts the RACF by sending it a Diameter Session Termination Request (STR) message. All ended calls must be deleted from the RACF in order to accurately track used and available bandwidth.

The RACF can apply its hosted policies for calls originating at SIP UAs located behind NATs. This is a standard part of the Oracle USM's ability to provide seamless HNT.



Bandwidth Negotiation

Because the decision whether to admit or reject a call is made before the INVITE is forwarded to the called party, some information is not available to the PDP at the initial request. The final IP Address, UDP port number, that transport the RTP flow, and the codec used are not known

by the Oracle USM until the called party responds with its own SDP information (either in the 180 or 200 response).

The Oracle USM examines the Session Description Protocol (SDP) value in the body of the SIP INVITE to determine what codecs are available for the call. If the INVITE specifies more than one codec, the Oracle USM bases its request to the RACF on the most bandwidth-hungry codec to ensure that all bandwidth requests will succeed or fail on the first attempt.

> **Note:**
>
> The amount of bandwidth requested depends on the configured media profiles.

If the call is admitted, and when the called party returns finalized SDP information, the Oracle USM modifies the original reservation with the chosen codec's bandwidth requirements. This ensures the RACF has current and accurate information with which to make policy decisions.

Session Lifetime

When receiving a successful Diameter response message for bandwidth from the RACF, a session lifetime timer may be included in the message. If included, this timer states how long the session can last. If the session continues past 3/4 of session lifetime, the Oracle USM sends another bandwidth request for that session to ultimately refresh the lifetime timer. If the RACF grants this bandwidth request, the Oracle USM continues to allow the session to proceed uninterrupted. If a lifetime timer for a session is not returned to the Oracle USM by the RACF, the Oracle USM assumes the session can last forever and never issues a refresh in this manner.

Opening for RTCP Flows

When the Oracle USM functions as the AF (i.e., A-SBC or P-CSCF), you can configure it to explicitly open RTCP ports, just as it does for RTP. Without explicitly opening these ports, the Oracle USM relies on possibly unreliable PDN-GWs and BRAs to open RTCP ports and it sends only RTP information in AARs.

In external bandwidth managements configurations where the application mode is Rx and the **include-rtcp-in-request** parameters is **enabled**, the Oracle USM ensures RTCP ports are opened. It sends AAR requests to the policy server (PCRF) that contain both RTP and RTCP, opening the gates (ports) for both RTP and RTCP flows. RTCP information in the AAR is the number of the RTP port plus one (RTP port + 1 = RTCP ports) for all sessions. Flow information for RTCP is part of a different Media-Sub-component AVP as the RTP, but under the same Media-Component-Description AVP. RTCP flow information will also include the Flow-Usage AVP with RTCP (1). The RTCP port is set to 0 when the RTP ports is also unknown and therefore set to 0.

RACF-only AVPs

Diameter AAR Query Post SDP Exchange

The Oracle USM supports sending the Authentication-Authorize-Request (AAR) query upon SDP answer instead of the SDP offer. This change can useful in WiMax environments where mobile phones go idle and become semi-detached from their base stations and from the WiMax

access controller (WAC). In such a case, the WAC receives an AAR from the idle user but, because it cannot determine that user's base station, rejects the request.

You enable this behavior by setting the **reserve-incomplete** parameter to **orig-realm-only**.

The Proxy Bit

When a signaling protocol receives an event request, the Oracle USM must ensure that the external policy server on the other end has enough bandwidth to maintain the requested call. The SDP information from the signaling message is stripped and encoded into the Diameter Band Request to be forwarded onto the external policy server. This feature is used with the Gq and other Diameter based interfaces.

The proxy bit allows the Oracle USM to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network. A parameter in the **ext-policy-server** configuration element called **allow-srv-proxy** has been developed. When this parameter is enabled, the proxy bit is set and the external policy server must process this Diameter request. When the parameter is disabled, the Oracle USM gives the external policy server permission to proxy the request along.

If you do not use this feature, this external policy server either handles the Diameter message on its own or proxies it to another server, depending on how much traffic it is handling at the time. This is done without any input from the Oracle USM.

Experimental-Result-Code AVP: RACF

The Diameter RACF interface takes special actions based upon what AVPs are present inside the Experimental-Result AVP in the User-Data-Answer (UDA) message. If the Experimental-Result-Code AVP found within an Experimental-Result AVP has any value that is not considered successful, per RFC 3588, the response will be handled as non-successful response and a 503 error code will be issued back to the endpoint. If the value is a successful one, the normal call flow will proceed.

Transport-Class AVP

When the Oracle USM, running as a Diameter-based RACF, receives a SIP INVITE triggering external bandwidth management, the Oracle USM performs SDP stripping and—through internal processes—selects an external bandwidth manager to use. If the options parameter in the selected external bandwidth manager is set to transport-class, the Oracle USM's Diameter RACF interface will issue authentication authorization requests (AARs) with the transport class AVP. The Oracle USM does not insert the transport-class AVP messages when the option is not configured.

The transport-class AVP will:

- Be identified with the AVP code 311
- Always have the vendor (V) bit set in the AVP flags
- Never have the mandatory (M) bit set in the AVP flags
- Have the Vendor-Id field set to 13019 (a value specified by ETSI TISPAN)
- Be formatted as an unsigned integer
- Reside in the Media-Component AVP, a grouped AVP

In addition, the transport-class AVP's payload field will be a 32-bit unsigned integer corresponding to a specific media type. The Oracle USM learns the specific media type from the m-line of the SDP it received. The following table shows how the Oracle USM evaluates the SDP's m= lines and concludes a default service type classification.

Service Class Classification	SDP evaluation	Default transport-class value
video	At least 1 m=video line	1
audio	No m= video At least 1m=audio	0
application	No m=video, No m=audio At least 1 m=application	3
data	No m=video, No m=audio, No m=application At least 1 m=data	2
control	No m=video, No m=audio, No m=application, No m=data At least 1 m=control	4
image	No m=video, No m=audio, No m=application, No m=data, No m=control At least 1 m=image	10
text	No m=video, No m=audio, No m=application, No m=data, No m=control, No m=image At least 1 m=text	5
message	No m=video, No m=audio, No m=application, No m=data, No m=control, No m= image, No m=text At least 1 m=message	6

After the service class classification has been chosen, the Oracle USM inserts the corresponding default transport class value in the transport-class AVP to send to the RACF.

Overriding Transport- Class AVP Value

You can override the Transport class AVP value sent to the RACF in a Diameter message by configuring the **service class options** parameter. Custom service class option values are configured as a <service-class-option>=<user-entered-value> value pair. For example, to configure the Oracle USM to send sending the value 80 instead of the value 8 for a message service class classification, you would configure **message=80** in the service class options parameter.

Transport-class AVP Configuration

You configure the Oracle USM to send the Transport-Class AVP in the external bandwidth manager's **options** parameter.

To set the transport-class AVP support for an external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. **options**—Set the options parameter by typing options, a Space, the option name **transport-class** with a plus sign in front of it. Then press Enter.

```
ORACLE(ext-policy-server)# options +transport-class
```

If you type **options** and then the option value without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. **service-class-options**—Set this options parameter by typing **service-class-options**, a <space>, the service-class type, an equal sign, your custom value. Then press Enter. Enter multiple service class/value pairs separated by a comma. For example:

```
ORACLE(ext-policy-server)# service-class-options message=80,text=70
```

6. Save and activate your configuration.

RACF and CLF AVPs

Frame-IP-Address AVP

The Diameter CLF and RACF interfaces can send a Frame-IP-Address AVP. You can configure the value to appear in either an ascii string (e.g., 192.168.10.1) or an octet string (e.g., 0xC0A80A01) with the **framed ip addr encoding** parameter.

1637 - Diameter Destination Realm AVP

As of S-C6.2.0, the Destination Realm AVP's value does not contain the realm of the incoming SIP message. Now, it contains the realm where the Policy Server resides as learned from the Origin-Realm AVP received in a CEA message from the Policy Server.

The Oracle USM can be configured with an option, **include-gua** , to retain the previous behavior of sending an incoming SIP message's realm to a policy server. This is accomplished by sending the Globally Unique AVP in the AAR message to the policy server, by adding an option parameter to the external policy server configuration.

The following table summarizes the effect of provisioning the external policy server with the Globally Unique AVP option on each Diameter interface, as configured.

Diameter Interface	No include-gua option Configured (default)	Add include-gua option
Rq	AAR sends Globally Unique Address AVP	AAR sends Globally Unique Address AVP
Rx	AAR does not send Globally Unique Address AVP	AAR will contain Globally Unique Address AVP
Gq	AAR does not send Globally Unique Address AVP	AAR will contain Globally Unique Address AVP

Diameter Interface	No include-gua option Configured (default)	Add include-gua option
E2	AAR sends Globally Unique Address AVP	AAR sends Globally Unique Address AVP

Legacy Destination-Realm AVP Behavior

The Diameter CLF and RACF interfaces can change the format of the payload string in the Destination-Realm AVP for any Diameter message it originates and sends to an external server. The payload field for this AVP can be constructed in any the following formats:

Format	Description
<user>@<realm>	<user>—IP address of the endpoint initiating the call with the Oracle USM <realm>—Name of the realm on which the Oracle USMreceived the INVITE from a user
<user>	<user>—IP address of the endpoint initiating the call with the Oracle USM
<realm>	<realm>—Name of the realm on which the Oracle USM received the INVITE from a user

When either the Diameter CLF or RACF interface sends any message with the Destination-Realm AVP, it determines from the external policy server configuration how to construct the payload string for this AVP.

You can set the format to use in the **dest-realm-format** parameter in the external policy server configuration. The parameter can be set to any value in the table above and defaults to <user>@<realm>. By treating the format this way, the policy server and the Oracle USM can easily communicate this value; if sent to the policy server in any AVP, the policy server can simply return the full value.

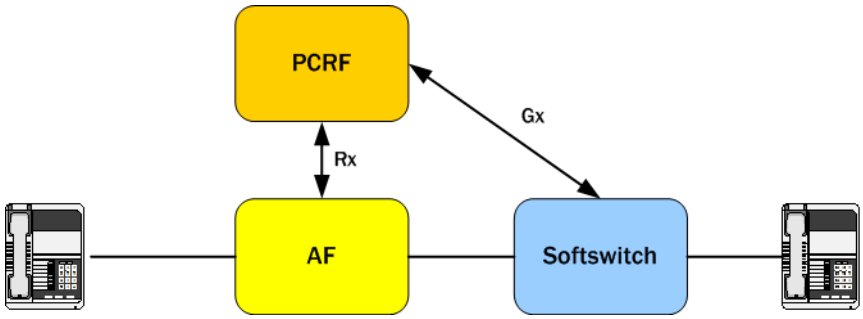
Origin-Host AVP

The Diameter CLF and RACF interfaces can change the suffix for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name.

You can set the suffix you want appended to payload strings using the **domain-name-suffix** parameter in the external policy server configuration. This parameter can be set to any string (default is .com), and the Oracle USM automatically adds a dot (.) to the front of this entry if you do not include one. The policy server and the Oracle USMcan easily communicate this value; if sent to the policy server in any AVP, the policy server can simply return the full value.

Wildcard Transport Protocol

The Oracle USM external bandwidth management solution provides for an Rx interface that supports the Flow-Description AVP (507). Rather than use a numerical value, this Flow-Description AVP uses an IP filter rule with the keyword “ip.” The ip keyword means any transport protocol matches the Flow-Description AVP when issuing AARs to the PCRF. Before it forwards a Gx RAR messages to the softswitch, the PCRF decodes the audio codec into the correct speed and classification. The PCRF passes the Oracle USMs Flow-Description AVP to the softswitch untouched. But not all softswitches accommodate the ip keyword, resulting in rejected requests.



When you enable the **wildcard-transport-protocol** parameter, however, you can essentially format the Flow-Description AVP to suit your network.

For sessions that need to allocate media and have applicable external bandwidth management associations, the Oracle USM's Diameter interface checks for the necessary bandwidth. The Diameter interface, with an Rx or Rq application mode, constructs an AAR with Flow-Description AVP of ip when the **wildcard-transport-protocol** parameter is **enabled**. The flow description would look like this:

```
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit out ip from 168.192.24.20 49500 to 168.192.24.0 8000
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit in ip from 168.192.24.0 8000 to 168.192.24.20 49500
```

With the **wildcard-transport-protocol** set to **disabled**, the Oracle USM does not use the ip wildcard keyword. Instead, it uses the specific media stream transport protocol in the Flow-Description AVP—and only falls by to the ip keyword when the transport protocol is unknown. The flow description with this parameter disabled would look like this:

```
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit out 17 from 168.192.24.20 49500 to 168.192.24.0 8000
<Flow-Description-AVP(507) | Avp Flags=128 | AVP Length=72 | Vendor-Id=10415
Data = permit in 17 from 168.192.24.0 8000 to 168.192.24.20 49500
```

New Configurations and Upgrading

To comply with 2GPP TS 29.213, **wildcard-transport-protocol** parameter is **disabled** by default in new configurations. So if the transport protocol is known, the Oracle USM uses it in the Flow-Description AVP.

To maintain default behavior for existing configurations, the Oracle USM performs a check at the time of upgrade to set this parameter to **enabled**. This setting means the Oracle USM does use the ip keyword in the Flow-Description AVP.

Configuring Diameter-based RACF

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle USM in the role of an Access N Oracle USM. In this example, you will configure additions to the ream configuration and the new external bandwidth manager configuration. You must also configure media profiles to accept bandwidth policing parameters.

Diameter Support Realm Configuration

To configure the realm configuration for Diameter support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

   ```
   ORACLE(realm-config)# select 1
   ORACLE(realm-config)#
   ```

5. **mm-in-realm**—Set this parameter to enabled so that calls from devices in the same realm have their media flow through the Oracle USM to be subject to CAC. The default value is **disabled**. The valid values are:

   - enabled | disabled

6. **mm-in-network**—Set this parameter to **enabled** so that the Oracle USM will steer all media traveling between two endpoints located in different realms, but within the same network. If this field is set to disabled, then each endpoint will send its media directly to the other endpoint located in a different realm, but within the same network. The default value is **enabled**. The valid values are:

   - enabled | disabled

7. **ext-bw-manager**—Enter the name of the external bandwidth manager configuration instance to be used for external CAC for this Realm.

8. Save your work using the ACLI **done** command.

External Bandwidth Manager Configuration

To configure the external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

4. **name**—Enter the name for this external bandwidth manager instance. This parameter is used to identify the PDP that will be used in each Realm configuration.

5. **state**—Set the state of this ext-policy-server configuration to **enabled** to run this CAC. The default value is **enabled**. The valid values are:

   - enabled | disabled

6. **operation-type**—Enter **bandwidth-mgmt** for this external policy server configuration element to perform RACF/External Policy Server functions. The default value is **disabled**. The valid values are:
7. **protocol**—Enter **Diameter** to support Diameter-based CAC. The default value is **C-SOAP**.
8. **address**—Enter the IP Address or FQDN of the external RACF.
9. **port**—Enter the port number the diameter connection connects to on the RACF. The default value is **80**. The valid range is:
   - Minimum—0
   - Maximum—65535
10. **realm**—Enter the name of the Realm in which this Oracle USM defines the RACF to exist. This is NOT necessarily the Realm where the Oracle USM performs admission control.
11. **permit-conn-down**—Enter **enabled** if this external policy server configuration can permit new calls into the network when the policy server connection is down. The default value is **disabled**. The valid values are:
12. **product-name**—Enter text string that describes the vendor-assigned name for the RACF. This parameter is required.
13. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**.
14. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the RACF. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
15. **framed-ip-addr-encoding**—Enter the format of the Frame-IP-Address (AVP 8) value in Diameter messages. The default value is **octet-string**. The valid values are:
    - **ascii-string**—Example: 192.168.10.1
    - **octet-string**—Example: 0xC0A80A01
16. **dest-realm-format**—Enter the format you want to use for the Destination-Realm AVP. The default value is **user_with_realm**. The valid values are:
    - user_with_realm | user_only | realm_only
17. **domain-name-suffix**—Enter the suffix you want to use for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name Your value can be any string, to which the system will prepend with a dot if you do not include one. The default value is **.com**.
18. **allow-srv-proxy**—Set to enabled in order to include the proxy bit in the header. The presence of the proxy bit allows the Oracle USM to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network (disabled). The default is enabled. The valid values are:
    - enabled | disabled
19. **wildcard-trans-protocol**—Set this parameter from **enabled** if you want to use transport protocol wildcarding for Rx/Rq Flow-Description AVP (507) generation. Enabled sends a flow description of ip. Set this parameter to **disabled** if you want to use the specific media stream transport protocol.

20. **reserve-incomplete**—Set this parameter to **enabled** when communicating with a PDP via Diameter. The parameter allows the Oracle USM to make admission requests before learning all the details of the flows and devices (e.g., not knowing the final UDP port numbers for the RTP media streams until after the RTP has begun). The default value is **enabled**. The valid values are:
    - **enabled** (default)—This mode supports the usual behavior when the AAR is sent upon SDP offer as well as SDP answer. This mode ensures backwards compatibility.
    - **orig-realm-only**—This mode allows calls originating from a realm with a policy server associated with it to send the AAR upon SDP offer. However, calls terminating at a realm with a policy server associated with it send the AAR post SDP exchange.
    - **disabled**—This mode allows no bandwidth reservation for incomplete flows.
21. **include-rtcp-in-request**—Change this parameter from **disabled** (default), to **enabled** so the Oracle USM will include RTCP information in AARs. RTCP information is the number of the RTP port plus one (RTP port + 1 = RTCP ports) for all sessions.
22. **trans-expires**—Set the amount of time, in seconds, that the Oracle USM waits before performing an expiration if a Diameter base protocol transaction does not occur. The default value is 15 seconds. Valid values range between 1 and 15.
23. Save your work using the ACLI **done** command.

Media Profile Configuration

Values for the following parameters can be found in the PacketCable™ Audio/Video Codecs Specification PKT-SP-CODEC-I06-050812 document.

To configure the media profile configuration for Diameter support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

    ```
    ORACLE# configure terminal
    ```

2. Type **session-router** and press Enter to access the session router path.

    ```
    ORACLE(configure)# session-router
    ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

    ```
    ORACLE(session-router)# media-profile
    ORACLE(media-profile)#
    ```

4. Type **select** and the number of the pre-configured media profile you want to configure.

    ```
    ORACLE(media-profile)# select 1
    ORACLE(media-profile)#
    ```

5. **req-bandwidth**—Enter the required bandwidth in Kbps for the selected media profile. This is the bandwidth that the Oracle USM will request from the policy server. The default value is zero (**0**). The valid values are:
    - Minimum—0
    - Maximum—4294967295
6. **standard-pkt-rate**—Enter the value to use for the standard packet rate for this codec when sending a request to the RACF for a bandwidth request.
7. Save your work using the ACLI **done** command.

CAC Debugging

A new argument has been added to the show command for viewing CAC statistics. From the user prompt, type **show ext-band-mgr**.

```
ORACLE# show ext-band-mgr
10:11:38-194
EBM Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                1       1       1          1       1       1
Connections            0       0       0          0       0       0
                        ---- Lifetime ----
                     Recent      Total  PerMax
Reserve                   0          0       0
Modify                    0          0       0
Commit                    0          0       0
Remove                    0          0       0
EBM Requests              0          0       0
EBM Installs              0          0       0
EBM Errors                0          0       0
EBM Rejects               0          0       0
EBM Expires               0          0       0
EBMD Errors               0          0       0
```

You can also refer to the log.ebmd log file located in the /ramdrv/logs/ directory on the Oracle USM. This file must be retrieved via FTP or SFTP.

2127 - Configurable Subscription ID Types

Subscriber Information AVP

Certain policy servers rely on having the user's URI information available as means to identify the endpoint/subscriber. In addition to conveying the L3 IP address of a user, the Oracle USM supports RFC 4006: The Subscription-Id AVP. It identifies the end user's subscription and is used in 3GPP Rx reference point. This feature can be enabled regardless of the selected application mode of the external policy server.

You can enable the Oracle USM to include the Subscription-ID-Type based on the received message’s request line’s Request URI, or you can set the Subscription ID type on a per-realm basis.

Subscription-ID AVP

The Subscription-Id AVP (AVP Code 443) includes a Subscription-Id-Data AVP that holds the identifier and a Subscription-Id-Type AVP that defines the identifier type. The external policy server configuration element is configured with an option to enable sending the Subscription-Id AVP to the policy server in an AA-Request (AAR) message.

Subscription-Id-Type

The Oracle USM can send a Subscription-Id-Types AVP to an external policy server:

Value	Name	Description
0	END_USER_E164	Identifier is in international E.164 format (e.g., MSISDN), according to the ITU-T E.164 numbering plan.
1	END_USER_IMSI	Identifier is in international IMSI format, according to ITU-T E. 212 numbering plan as defined in [E212] and [CE212].
2	END_USER_SIP_URI	Identifier is in the form of a SIP/SIPS URI.

To send the Subscription-ID-Types AVP in an AAR, you must add the include-sub-info option in the external policy server configuration element. If this option is enabled, and you do not configure a **subscription-id-type** in the realm-config, the request line's Request URI of the received message is sent in the AAR.

You can also set the Subscription-ID-Type on a per-realm basis by configuring the parameter **subscription-id-type** in the realm-config element to determine the value of the Subscription ID in the AAR.

The default value for **subscription-id-type** is END_USER_NONE. If the value of subscription-id-type is set to END_USER_NONE, and the external policy server option is enabled to send the Subscription ID AVP, then the Oracle USM relies on the request line's Request URI of the received message.

If the value for **subscription-id-type** is set to one format, such as END_USER_IMSI, but the request line's Request URI is set to another format, such as END_USER_SIP_URI, the value of **subscription-id-type** is used to send in the AAR message.

Subscription ID AVP in AA-Request Message Configuration

To configure the Oracle USM to send Subscription ID AVP in AA-Request Messages:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **ext-policy-server** and press Enter.

   ```
   ORACLE(session-router)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the external policy server that you want to edit.

4. **options**—Set the options parameter by typing **options**, a Space, the option-name **include-sub-info** with a plus sign in front of it, and then press Enter.

   ```
   ORACLE(ext-policy-server)# options +include-sub-info
   ```

   If you type the option without the plus sign, you will overwrite any previously configured options. In order to append the new options to the policy server configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Subscription ID AVPs in AA-Request Message Configuration

To set the Subscription ID AVPs in AA-Request Messages:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(session-router)# realm-config
   ORACLE(realm-config)#
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI select command) the realm that you want to edit.

4. **subscription-id-type**—Set the Subscription ID Type included in the AA-Request.
   - END_USER_E164
   - END_USER_IMSI
   - END_USER_SIP_URI
   - END_USER_NONE

5. Save and activate your configuration.

Specific Action AVP support

When acting as a P-CSCF, Oracle USM sends the Specific-Action AVP to the PCRF in an AAR message to indicate the subscription types it supports.

The Oracle USM can be configured to subscribe to one or more of the following subscription types:

- LOSS OF BEARER
- RECOVERY OF BEARER
- RELEASE OF BEARER
- OUT OF CREDIT
- SUCCESSFUL RESOURCES ALLOCATION
- FAILED RESOURCES ALLOCATION

When no subscription types are configured, the Oracle USM does not include the Specific-Action AVP in its AAR.

Specific Action AVP subscription is configured in the **Specific action subscription** parameter located in the External policy server configuration element.

ACLI Examples - specific-action-subscription

1. Navigate to the ext-policy-server configuration element.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# ext-policy-server
ACMEPACKET(ext-policy-server)#
```

2. Select an existing external policy server configuration element.

```
ACMEPACKET(ext-policy-server)# select
name:
1: extpol1

selection: 1
ACMEPACKET(ext-policy-server)#
```

3. specific-action-subscription - Configure 1 or more specific actions. When configuring 2 or more specific actions, enclose them in quotation marks, with the values separated by spaces. The following are valid specific actions: loss-of-bearer, recovery-of-bearer, release-of-bearer, out-of-credit, successful-resources-allocation, failed-resources-allocation

```
ACMEPACKET(ext-policy-server)# specific-action-subscription "loss-of-bearer
recovery-of-bearer"
```

4. Type done to save your work.
5. Save and activate your configuration to begin using this feature.

Diameter STR Timeouts

When a call ends, the Oracle USM alerts the RACF by sending it a Diameter Session Termination Request (STR) message. You can enable the Oracle USM to resend STR messages at the application layer if the STR messages time out. A new option **STR-retry=x** has been created that allows you to configure the number of times the STR messages are resent.

Diameter STR Timeouts Configuration

To configure Diameter request timeouts:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
ORACLE(media-manager)#
```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

4. Set the options parameter by typing **options**, a Space, the option name **STR-retry=x** with a plus sign in front of it. Then press Enter.

```
ORACLE(ext-policy-server)# options +STR-retry=x
```

If you type options and then the option value without the plus sign, you will overwrite any previously configured options. In order to append the new options to this configuration's options list, you must prepend the new option with a plus sign as shown in the previous example.

5. Save and activate your configuration.

Gq Interface Features

The Oracle USM can run the Gq interface over a Diameter connection and act as a P-CSCF (AF) communicating with a PDF. The application ID field must be set to 16777222 to run the Gq reference point.

Rx Interface Features

The Oracle USM can run the Rx interface over a Diameter connection and act as a P-CSCF communicating with a PCRF. The application ID field must be set to 16777236 to run the Rx reference point. When running in this mode, the Oracle USM will send two AVPs, in addition to other information:

- The Codec-Data AVP is then included for non-priority calls. This AVP is one of several that together comprise a Group AVP structure.
- The Reservation-Priority AVP is included for priority calls. This AVP will be the main AVP within the AAR message.

Non-Priority Call Handling

When a SIP signaling event triggers external bandwidth management use, the Oracle USM removes all SDP information from the signaling message that was the trigger. The Oracle USM repackages this bandwidth information so that it can form a Bandwidth Request and decide on an external bandwidth manager to which it should be sent. If the appropriate external bandwidth manager is configured for Rx interface use, then Oracle USM then reformats the SDP information to construct a Codec-Data AVP.

If the external bandwidth manager that receives the request ignores the SDP information, then it does not include the Codec-Data AVP in the AAR.

For calls that do not require special treatment, the Codec-Data AVP is required to have the:

- AVP code 599
- 3GPP vendor identification number (10415)
- V (vendor) bit set in the AVP
- M (mandatory) bit set when sending this AVP
- Type octet string

In addition, the Codec-Data AVP appears as described in the following table.

AVP section/line	Requirement
Line 1	Must specify the direction of the flow by including the ASCII “uplink” or downlink: uplink—Identifies the SDP as having come from the UE and sent to the network downlink—Identifies the SDP as having come from the network and sent to the UE
Line 2	Must specify whether the offer or answer codec is at issue by including the ASCII “offer” (from an SDP offer according to RFC 3264) or answer (from an answer according to RFC 3264)

AVP section/line	Requirement
Remainder of the AVP	Must include lines found in the signaling SDP, formatted in ASCII and separated by new-line characters; the first line of this section must be the m line, followed by any “a” or “b” lines related to that m line

Priority Call Handling

The Oracle USM determines that a call is priority call when it matches a defined network management control (NMC) priority rule. No other scenario triggers the priority call handing treatment described in this section.

When a SIP signaling event triggers external bandwidth management use for a priority call, the Oracle USM sends the Reservation-Priority AVP in the AAR message. The Reservation-Priority AVP is required to:

- Use the ETSI Vendor identification number (13019)
- Have the V (vendor) bit set in the AVP
- Not to have the M (mandatory) bit set when sending this AVP
- Be of type enumeration
- Set to PRIORITY-SEVEN (7)

Rx bearer plane event

The Rx reference point is used to exchange application level session information between the PCRF and the AF which is the Oracle P-CSCF/SBC. The PCRF exchanges the Policy and Charging Control (PCC) rules with the PCEF (Policy and Charging Enforcement Function) and QoS rules with the BBERF (Bearer Binding and Event Reporting Function). The role of the former is typically performed by a 3GPP/GGSN, 3GPP-R9/PGW, 3GPP2/PDSN, WiFi/Max/ASN-GW. The role of the latter may be a standalone element or typically it’s incorporated within the system acting as the PCEF. The Oracle P-CSCF/SBC offers applications that require policy and charging control of traffic plane resources (e.g. UMTS PS domain/GPRS domain resources). Currently support for Rx within the Oracle USM supports policy, the present requirement will enhance Rx to incorporate traffic plane event notifications north bound. The PCRF receives session and media related information from the Oracle USM currently and with this enhancement the PCRF will report traffic plane events to the Oracle USM.

BN-Bearer-0040: When not all the service data flows within the AF session are affected, the PCRF will send an RAR command that includes the deactivated IP flows encoded in the Flows AVP and the cause encoded in the Specific-Action AVP.

BN-Bearer-0050: The Oracle SBC/P-CSCF shall acknowledge the receipt of this RAR command by sending an RAA. It shall then proceed to release the entire session at the SIP signaling layer and proceed to send the corresponding Diameter STR command to the PCRF and receive the STA.

AA-Request (AAR) Command

The following Specific-Action notifications will be added to the AA-Request command:

- INDICATION_OF_LOSS_OF_BEARER (2)
- INDICATION_OF_RECOVERY_OF_BEARER (3)

- INDICATION_OF_RELEASE_OF_BEARER (4)
- INDICATION_OF_OUT_OF_CREDIT (7)
- INDICATION_OF_SUCCESSFUL_RESOURCES_ALLOCATION (8)
- INDICATION_OF_FAILED_RESOURCES

Re-Auth-Request (RAR) Command Handling

The Flows AVP (510) and Abort-Cause AVP (500) will be ignored if they are received in the RAR command.

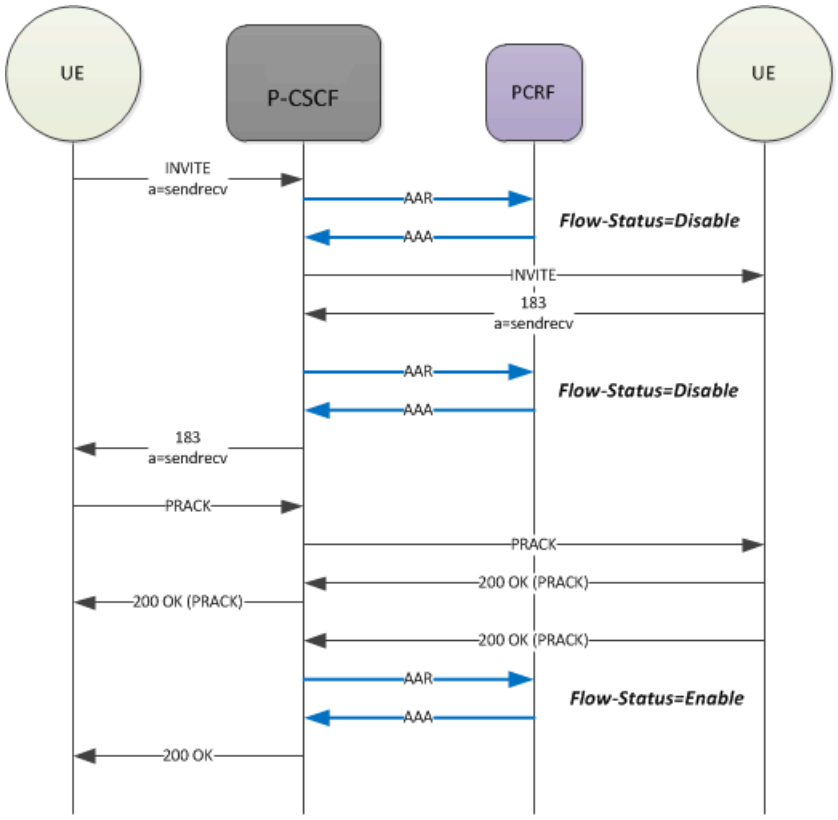
The Specific-Action AVP (513) will be handled as follows:

- Notification: INDICATION_OF_LOSS_OF_BEARER (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_RELEASE_OF_BEARER (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_OUT_OF_CREDIT(Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_FAILED_RESOURCES_ALLOCATION (Action: Terminate SIP Session * Send RAA BYE will be sent to both the EU and the core. This is similar to the Abort-Session--Request (ASR) handling).
- Notification: INDICATION_OF_SUCCESSFUL_RESOURCES_ALLOCATION (Action: Send RAA).
- Notification: INDICATION_OF_RECOVERY_OF_BEARER (Action: Send RAA)

2836 - Early Media Suppression for Rx

The Oracle USM supports early media suppression where it does not allow media to flow between the endpoints it connects until the session is fully established as noted by receiving a 200 OK. Early media supression is configured by setting the **early-media-allow** parameter on a realm, realm group, or session agent to **none**.

The Oracle USM also relays the state of early media suppression in the Flow-Status AVP in an AAR message to the PCRF as the call is being set up. When the call's media is still in a suppressed state, the Flow-Status AVP is set to Disabled. The following image illustrates when the Flow-Status changes from disable to enable.



Media-Component-Number and Flow-Number AVP Values

The values assigned to the Media-Component-Number and Flow-Number AVPs are not derived in compliance with 3GPP standards. Version S-CZ7.2.0 provides the capability to derive compliant values.

AA-Request (AAR) messages are sent, via the Rx Interface, by an Access Function (AF) to a Policy and Charging Rules Function (PCRF) in order to provide it with the session information. Among other fields, these messages contain a Media-Component-Number AVP and a Flow-Number AVP, used to identify specific media sessions, and the individual IP flows associated with these sessions.

The values assigned to the Media-Component-Number and Flow-Number AVPs are not derived in compliance with 3GPP standards.

Release S-CZ7.2.0 provides a new option that enables compliance with 3GPP requirements. Note that this support is limited to the Rx Interface

Media-Component-Number AVP

The Media-Component-Number AVP (AVP code 518) is of type Unsigned32; it contains the ordinal number of the media component (the SDP m= line) as specified in 3GPP TS 29.214. When this AVP refers to AF signalling, it contains the value 0. The Media-Component-Number AVP provides an index to the grouped Media-Component-Description AVP (AVP code 517).

3GPP compliance requires that:

1. Number should start from 1

2. It should contain the ordinal number of the position of the m= line in the SDP.
3. When this AVP refers to AF Signalling, this is indicated by using the value 0.
4. The ordinal number of a media component shall not be changed when the session description information is modified.

Flow-Number AVP

The Flow-Number AVP (AVP code 509) is of type Unsigned32; it contains the ordinal of the IP flows created to support a specific SDP m= line as specified in 3GPP TS 29.214. The Flow-Number AVP provides an index to the grouped Media-Sub-Component AVP (AVP code 519).

3GPP compliance requires that:

1. Number should start from 1
2. It should contain the ordinal number of the IP flow(s) within the m= line assigned in the order of increasing downlink destination port numbers, if downlink destination port numbers are available.
3. It should contain the ordinal number of the IP flow(s) within the m= line assigned in the order of increasing uplink destination port numbers, if no downlink destination port numbers are available.
4. The ordinal number assigned should not be changed when the session description information is modified.

Flow Examples

The following example illustrates the derivation of flow identifiers from SDP as specified by 3GPP TS 29.214 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Policy and Charging Control over Rx reference point (Version 9.3.0).

A flow identifier consists of two digits in the form #,# with the digit identifying a specific SDP m= line (defining a media session), and the second digit identifying a specific IP flow (for example, an RTP stream) supporting the media session. 3GPP standards require that the Media-Component-Number AVP be populated with the first digit if the flow identifier and the Flow-Number AVP be polulated with the second digit.

A UE, as the offerer, sends a SDP session description (relevant sections shown below) to an application server.

```
v=0
o=ecsreid 3262464865 3262464868 IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
s=MM01
i=One unidirectional audio media and one unidirectional video media and one
bidirectional application
media
t=3262377600 3262809600
m=video 50230 RTP/AVP 31                          Ordinal 1 (3 media flows, 1
RTP & 2 RTCP)
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
a=recvonly
m=audio 50330 RTP/AVP 0                           Ordinal 2 (3 media flows, 1
RTP & 2 RTCP)
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
a=sendonly
m=application 50430 udp wb                        Ordinal 3 (2 media flows, up &
down links)
```

```
c=IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
a=sendrecv
```

The application returns (relevant sections):

```
v=0
o=ecsreid 3262464865 3262464868 IN IP6 2001:0646:00F1:0045:02D0:59FF:FE14:F33A
s=MM01
i=One unidirectional audio media and one unidirectional video media and one
bidirectional application
media
t=3262377600 3262809600
m=video 51372 RTP/AVP 31                          Ordinal 1
c=IN IP6 2001:0646:000A:03A7:02D0:59FF:FE40:2014
a=sendonly
m=audio 49170 RTP/AVP 0                           Ordinal 2
c=IN IP6 2001:0646:000A:03A7:02D0:59FF:FE40:2014
a=recvonly
m=application 32416 udp wb                        Ordinal 3
c=IN IP6 2001:0646:000A:03A7:0250:DAFF:FE0E:C6F2
a=sendrecv
```

From this offer–answer exchange of SDP parameters the UE and the application server each creates a list of flow identifiers as shown below.

Order of m= line	Type of IP Flows	Destination IP Address/Port Number of Flows	Flow Identifier
1	RTP (Video) DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50230	<1,1>
1	RTCP DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50231	<1,2>
1	RTCP UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 51373	<1,2>
2	RTP (Audio) UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 49170	<2,1>
2	RTCP DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50331	<2,2>
2	RTCP UL	2001:0646:000A:03A7:02D0:59FF:FE40:2014 / 49171	<2,2>
3	UDP (application) DL	2001:0646:00F1:0045:02D0:59FF:FE14:F33A / 50430	<3,1>
3	UDP (application) UL	2001:0646:000A:03A7:0250:DAFF:FE0E:C6F2 / 32416	<3,1>

Media-Component AVP 3GPP Compliance Configuration

Use this procedure to enable compliance with 3GPP requirements. Compliance is enabled with the **format-flow-id** option available in **ext-policy-server** configuration mode.

1. Access the **ext-policy-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# ext-policy-server
ORACLE(ext-policy-server)#
```

2. Select the **ext-policy-server** object to edit.

```
ORACLE(ext-policy-server)# select
<name>:1:  name=extpol1

selection: 1
ORACLE(ext-policy-server)#
```

3. **format-flow-id**—Add this option to enable 3GPP compliance.

```
ACMEPACKET(ext-policy-server)# options +format-flow-id
```

4. Type **done** to save your configuration.

Rx Interface Reason Header Usage

The Oracle USM has increased capability to map network events to a configurable SIP Reason header. The extended capability provides for the mapping of specific disconnect events on the Rx Interface.

Release S-CZ7.2.0, and later releases expands the usage of the Reason header and its associated parameters to include mapping of disconnect events on the Rx interface between an SBC (functioning in the P-CSCF role) and a Policy and Charging Rules Function (PCRF). With the expanded capability enabled, contents of the Reason header are based on either the contents of the Specific-Action AVP in Re-Authorization Request (RAR) messages issued by the PCRF, or the contents of the Abort-Cause AVP in Abort Session Request (ASR) messages issued by the PCRF.

To use this feature, set the **session-router** > **sip-interface** > **rx-sip-reason-mapping** parameter to enabled.

Specific-Action AVP

The Specific-Action AVP (AVP code 513) is of type Enumerated.

Within a PCRF initiated Re-Authorization Request, the Specific-Action AVP determines the type of the action.

The following reported disconnected events can be mapped to SIP Reason headers using the corresponding value to configure in a **local-response-map-entry** > **local-error**.

Specific-Action AVP value	**local-error configuration**
2 -- INDICATION_OF_LOSS_OF_BEARER	rx-rar-loss-of-bearer
4 -- INDICATION_OF_RELEASE_OF_BEARER	rx-rar-release-of-bearer
7 -- INDICATION_OF_OUT_OF_CREDIT	rx-rar-out-of-credit
9 -- INDICATION_OF_FAILED_RESOURCES_ALLOCATION	rx-rar-failed-resources-allocation

Abort-Action AVP

The Specific-Action AVP (AVP code 500) is of type Enumerated.

Within a PCRF initiated Abort Session Request (ASR), the Abort-Cause AVP identifies the cause of the ASR.

The following reported disconnected events can be mapped to SIP Reason headers using the corresponding value to configure in a **local-response-map-entry** > **local-error**.

Abort-Cause AVP value	local-error configuration
0 -- BEARER_RELEASED	rx-asr-bearer-released
1 -- INSUFFICIENT_SERVER_SERVICES	rx-asr-insufficient-server-resources
2 -- INSUFFICIENT_BEARER_SERVICES	rx-asr-insufficient-bearer-resources
3 -- PS_TO_CS_HANDOVER	rx-asr-ps-to-cs-handover
4 -- SPONSORED_DATA_CONNECTIVITY_DISALLOWED	rx-asr-sponsored-data-connectivity-disallowed

Message Flows

Mapped headers will appear in either a BYE or a CANCEL message from the P-CSCF. As shown in the following illustrations, a BYE message is issued when the disconnect event occurs after establishing the SIP session. A CANCEL message is issued when the disconnect event occurs before establishing the SIP session.

RAR Loss-of-Bearer After Session Establishment

If mapping is not enabled, the Reason header defaults to SIP;cause=503;text="Service Unavailable”.

Network Provided Location Information

Network provided location information (NPLI) is a service commonly supported within the IMS network architecture. NPLI is most commonly provided as a geographic identifier which specifies a location either in terms of geodetic coordinates (latitude and longitude), for example 42°25'33 N 71°18'16 W; or in terms of a recognized civic identifier, for example Lincoln, Massachusetts.

Support for NPLI was introduced in two phases. Initial, Phase 1 support for NPLI, was provided by Version S-CZ7.1.2. Enhanced, Phase 2 NPLI support, is provided by Version S-CZ7.2.0 and later releases.

Basic Implementation

Basic Phase 1 NPLI support is provided with Version S-CZ7.2.0.

While implementation details and the granularity of provided services differ among IMS providers, Phase 1 NPLI implementation triggered by an originating mobile device typically involves a message exchange similar to the following.

1. Upon receiving the initial INVITE from the mobile handset, the P-CSCF sends an AA-Request (AAR) command to the Policy Charging/Rule Function (PCRF) via the Rx interface.
2. The P-CSCF buffers the INVITE pending the receipt of an AA-Answer (AAA) command from the PCRF. Assuming NPLI is provided by the IMS core, the AAA contains a 3GPP-User-Location-Info Attribute Value Pair (AVP) along with a RAT-Type AVP. Depending upon network equipment, the AAA may also contain a proprietary MSISDN (Mobile Subscriber Integrated Services Digital Network Number), AVP although this is not usually the case.
3. Upon receiving the AAA, the P-CSCF, using values contained in the 3GPPUser-Location-Info attribute, and the RAT-Type attribute, adds an abbreviated P-Access-Network-Info (PANI) header to the buffered INVITE. If an MSISDN is included within the AAA, the P-CSCF also adds a P-Subscription-MSISDN header to the buffered INVITE.
4. The P-CSCF caches received attribute values, and forwards the altered INVITE to the IMS core.
5. All subsequent re-INVITES and UPDATES contain the PANI and P-Subscription-MSISDN headers populated with the cached values.

NPLI provisioning triggered by a terminating mobile device typically involves a message exchange similar to the following.

1. Upon receiving the initial INVITE from the IMS core, the P-CSCF sends an AAR command to the PCRF via the Rx interface.
2. The P-CSCF buffers the INVITE pending the receipt of an AAA command from the PCRF. Assuming NPLI is provided by the IMS core, the AAA contains the 3GPP-User-Location-Info and RAT-Type AVPs. Depending upon network equipment, the AAA may also contain a proprietary MSISDN AVP although this is not usually the case.
3. Upon receiving the AAA, the P-CSCF, using values contained in the 3GPPUser-Location-Info attribute, and RAT-Type attributes, adds a PANI header to the buffered INVITE. If an MSISDN is included within the AAA, the P-CSCF also adds a P-Subscription-MSISDN to the buffered INVITE.
4. The P-CSCF caches received attribute and MSISDN values, and forwards the altered INVITE to the mobile handset.
5. All subsequent 1xx and 200OK, and all subsequent re-INVITES and UPDATES that are generated by the mobile handset contain the PANI and P-Subscription-MSISDN headers populated with the cached values.

Enhanced Implementation

Enhanced Phase 2 NPLI support is provided by Version S-CZ7.2.0, and later releases. Phase 2 provides for an explicit subscription to NPLI changes resulting in the dynamic update of NPLI as opposed to the single static transaction provided by the initial Phase 1.

Phase 2 NPLI implementation triggered by an originating mobile device typically involves a message exchange similar to the following.

1. Upon receiving the initial INVITE from the mobile handset, the P-CSCF sends an AAR command to the PCRF via the Rx Interface. This AAR contains a Required-Access-Info AVP that requests user location reports as they become available to the PCRF.
2. The P-CSCF buffers the INVITE pending the receipt of an AAA command from the PCRF. The AAA contains a 3GPP-User-Location-Info AVP along with a RAT-Type AVP. Depending upon network design, the AAA may also contain a proprietary MSISDN AVP, although this is not usually the case.
3. Upon receiving the AAA, the P-CSCF, using values contained in the 3GPP-User-Location-Info attribute, and the RAT-Type attribute, adds an extended PANI header to the buffered INVITE. If an MSISDN is included within the AAA, the P-CSCF also adds a P-Subscription-MSISDN header to the buffered INVITE.
4. The P-CSCF caches received attribute values, and forwards the altered INVITE to the IMS core.
5. At any point during the call (from initial session establishment to session termination) the PCRF can send a Re-Auth Request (RAR) that contains updated NPLI conveyed by the 3GPP-User-Location-Info, RAT-Type, IP-CAN-Type, or MSISDN AVPs.
6. After receiving the RAR, the P-CSCF updates the cached attribute values.
7. All subsequent re-INVITES and UPDATES contain the PANI and P-Subscription-MSISDN headers populated with the most recent cache values.

The following call flow depicts a successful, mobile-originated call containing a Re-INVITE. NPLI updates are requested in the initial AAR (P-CSCF-to-PCRF). NPLI is first provided with the initial AAA (PCRF-to-P-CSCF). Updated NPLI is provided with the initial RAR (PCRF-to-P-CSCF). All requests/responses toward the IMS core contain the latest available NPLI.

3GPP-User-Location-Info Attribute

Phase I Implementation

The 3GPP-User-Location-Info Attribute (sub-attribute number 22) is defined in section 16.4.7.2, Coding 3GPP Vendor-Specific RADIUS attributes, of 3GPP TS29.061, Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN). This AVP contains the location type, for example a cell global identifier (CGI), and the actual location data for the specified cell.

Phase II Implementation

The 3GPP-User-Location-Info AVP (type 22) is defined in section 16.4.7.2, Coding 3GPP Vendor-Specific RADIUS attributes, of 3GPP TS29.061, Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN).

3GPP-User-Location-Info contains the location type, for example a cell global identifier (CGI), and the actual location data for the specified cell.

This attribute is supported by both the Phase 1 and Phase 2 implementations. It occurs in AAR and RAR messages received by the P-CSCF.

The AVP consists of four fields.

The 1-octet 3GPP-type field contains the value 22.

The 1-octet 3GPP-length field contains the attribute length in octets.

The 1-octet Geographic Location Type field identifies the equipment type whose location is being provided.

```
  0 = CGI
  1 = SAI
  2 = RAI
128 = TAI
129 = ECGI
130 = TAI and ECGI
```

Other values are reserved for future use.

The variable length Geographic Location field contains the geographic coordinates of the equipment specified in the Geographic Location Type field.

IP-CAN-Type AVP (Phase 2)

The IP-CAN-Type AVP (type 1027) specifies the type of Connectivity Access Network to which the user is connected.

This attribute is supported by the Phase 2 implementation. It occurs in RAR messages received by the P-CSCF.

The following values are defined.

```
0 = 3GPP-GPRS
    indicates 3GPP General Packet Radio Service (GPRS) access and is further
detailed by the RAT-Type AVP that contains an applicable value (not EUTRAN).
1 = DOCSIS
    indicates Data Over Cable Service Interface Specification (DOCSIS) access.
2 = xDSL
    indicates Data Over Cable Service Interface Specification (DOCSIS) access.
3 = WiMAX
    indicates Worldwide Interoperability for Microwave Access (WiMAX -- IEEE
802.16).
4 = 3GPP2
    indicates CDMA2000 access as specified in 3GPP2 X.S0011 and is further
detailed by the RAT-Type AVP.
5 = 3GPP-EPS
    indicates 3GPP Evolved Packet System (EPS) access and is further detailed by
the RAT-Type AVP.
6 = Non-3GPP-EPS
    indicates Evolved Packet System (EPS) based on non-3GPP access technology
and is further detailed by the RAT-Type AVP.
```

MSISDN AVP

The Vodafone proprietary MSISDN AVP is supported in both NPLI Phases 1 (Release S-CZ7.1.2) and 2 (Release s-CZ7.2.0 and later).

This attribute can be found in AAR and RAR messages received by the P-CSCF. It is constructed as follows.

```
<Name>MSISDN-VF:21274</Name>
<AvpName>MSISDN-VF</AvpName>
<AvpCode>3701</AvpCode>
<VendorId>21274</VendorId>
```

```
<MandatoryFlag>false</MandatoryFlag>
<ProtectFlag>false</ProtectFlag>
<MayencryptFlag>false</MayencryptFlag>
<VendorSpecificFlag>true</VendorSpecificFlag>
<AvpType>utf8String</AvpType>
```

RAT-Type Attribute

Phase I Implementation

The RAT-Type Attribute (attribute type 1032) specifies the radio access technology type. For purposes of NPLI, the following values are supported:

- 1000 — Universal Terrestrial Radio Access Network (UTRAN)
- 1001 — SMS Edge Radio Access Network (GERAN)
- 1004 — Evolved UMTS Radio Access Network (EUTRAN)

In the event that a received RAT-Type attribute contains an unsupported type (something other that 1000, 1001, or 1004), an error condition is declared and the default PANI is used.

Phase II Implementation

This attribute is supported by both the Phase 1 and Phase 2 implementations.

It occurs in AAR and RAR messages received by the P-CSCF.

The following values are supported.

```
   0 - WLAN
   1 - Virtual (indicates that RAT is unknown)
1000 – Universal Terrestrial Radio Access Network (UTRAN)
1001 – GSM Edge Radio Access Network (GERAN)
1002 - GAN (Generic Access Network)
1003 - HSPA_EVOLUTION (Evolved High Speed Packet Access)
1004 – Evolved UMTS Radio Access Network (EUTRAN)
2000 - CDMA2000_1X (core CDMA2000 wireless air interface)
2001 - HRPD (High Rate Packet Data)
2002 - UMB (Ultra Mobile Broadband)
2003 - EHRPD (Enhanced High Rate Packet Data)
```

In the event that a received RAT-Type attribute contains an unsupported type, an error condition is declared and the default PANI is used.

NPLI Required-Access-Info AVP (Phase 2)

The Required-Access-Info AVP (type 536) conveys a subscription request for NPLI.

This attribute is supported by the Phase 2 implementation. It occurs in AAR messages originated by the P-CSCF.

The following values are defined.

```
0 - subscribe to NPLI
1 - subscribe to user timezone information
```

NPLI Specific-Action AVP (Phase 2)

The Specific-Action AVP (type 513) works in conjunction with the Required-Access-Info AVP (type 536) to convey an NPLI subscription request to the PCRF.

This attribute is supported by the Phase 2 implementation. It occurs in AAR messages originated by the P-CSCF.

The following value is used to request NPLI:

```
12 - ACCESS_NETWORK_INFO_REPORT
```

NPLI Enabling NPLI Notifications

The P-CSCF sends the PCRF AAR messages containing the Specific-Action and Required-Access-Info AVPs to indicate a desired notification/subscription type. Use the following procedure to request NPLI notifications.

1. Move to ext-policy-server configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# media-manager
   ACMEPACKET(media-manager)# ext-policy-server
   ACMEPACKET(ext-policy-server)#
   ```

2. Use the **select** command to identify the target PCRF.

   ```
   ACMEPACKET(ext-policy-server)# select
   name:
   ...
   ...
   ...
   4. PCRF_10A
   ...
   ...
   ...
   selection: 4

   ACMEPACKET(ext-policy-server)#
   ```

3. Use the **specific-action-subscription** parameter to specify one or more specific actions. When designating 2 or more actions, enclose them in quotation marks, with the values separated by spaces. Use **access-network-info-report** to enable NPLI support.

   ```
   ACMEPACKET(ext-policy-server)# specific-action-subscription
   access-network-info-report
   ACMEPACKET(ext-policy-server)#
   ```

4. Use **done**, **exit** and **verify-config** to complete configuration.

P-Access-Network-Info SIP Header

The P-Access-Network-Info SIP header is defined in section 4.4 of RFC 3455, *Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)*. This header contains information on the access network that the user handset is using to get IP connectivity, and is typically ignored by intermediate proxies between the SBC that adds the header and the SIP proxy that is providing services. The proxy providing services can inspect the header and make use of the information contained there to

provide appropriate services, depending on the value of the header. Before proxying the request onwards, this proxy strips the header from the message.

The P-Access-Network-Info header contains an access-type field identifying the radio access technology as received in the RAT-Type attribute.

- 3GPP-UTRAN — when the RAT- Type AVP value is 1000
- 3GPP-GERAN — when the RAT- Type AVP value is 1001
- 3GPP-E-UTRAN — when the RAT- Type AVP value is 1004

The access-type field is followed by the access-info field which identifies the mobile phone cell that provides network access. The location of this cell provides the means to geographically locate connected mobile handsets.

- utran-cell-id-3gpp — when the RAT- Type AVP value is 1000
- cgi-3gpp — when the RAT- Type AVP value is 1001
- utran-cell-id-3gpp — when the RAT- Type AVP value is 1004

The access-info field is followed by an EQUAL sign and a CellID/CGI (Cell Global Identifier) field that contains an octet string which provides the location value contained in the 3GPP-User-Location-Info AVP. The location value can be either an CGI or an Extended CGI (ECGI).

- CGI field — when the RAT- Type AVP value is 1000
- CGI field — when the RAT- Type AVP value is 1001
- ECGI field— when the RAT- Type AVP value is 1004

Examples of P-Access-Network-Info headers are as follows:

If the RAT-Type is UTRAN:

P-Access-Network-Info: 3GPP-UTRAN;utran-cell-id-3gpp=C359A3913B20E

If the RAT-Type is GERAN:

P-Access-Network-Info: 3GPP-GERAN;cgi-3gpp=62F8100005C599

If the RAT-Type is EUTRAN:

P-Access-Network-Info: 3GPP-E-UTRAN;utran-cell-id-3gpp=C359A3913B20E

Abbreviated PANI SIP Header (Phase 1)

NPLI Phase 1 (Version S-CX7.1.2) provides an abbreviated PANI header consisting of three fields -- access-type, access-info, and location-info.

The access-type field identifies the radio access technology as identified by the RAT-Type attribute.

```
3GPP-UTRAN   – when the RAT-Type AVP value is 1000
3GPP-GERAN   – when the RAT-Type AVP value is 1001
3GPP-E-UTRAN – when the RAT-Type AVP value is 1004
```

The access-info field works in conjunction with the location-info filed to identifies the type and location of the mobile phone cell that currently provides network access. The location of this cell provides the means to geographically locate connected mobile handsets.

```
utran-cell-id-3gpp – when the RAT-Type AVP value is 1000
cgi-3gpp           – when the RAT-Type AVP value is 1001
utran-cell-id-3gpp – when the RAT-Type AVP value is 1004
```

The access-info field is followed by an EQUAL sign and a CellID/CGI (Cell Global Identifier) field that contains an octet string which provides the location value contained in the 3GPP-User-Location-Info AVP. The location value can be either an CGI or an Extended CGI (ECGI).

```
CGI field  – when the RAT-Type AVP value is 1000
CGI field  – when the RAT-Type AVP value is 1001
ECGI field – when the RAT-Type AVP value is 1004
```

Examples of abbreviated headers are as follows.

If the RAT-Type is UTRAN:

```
P-Access-Network-Info: 3GPP-UTRAN;utran-cell-id-3gpp=C359A3913B20E
```

If the RAT-Type is GERAN:

```
P-Access-Network-Info: 3GPP-GERAN;cgi-3gpp=62F8100005C599
```

If the RAT-Type is EUTRAN:

```
P-Access-Network-Info: 3GPP-E-UTRAN;utran-cell-id-3gpp=C359A3913B20E
```

Extended PANI SIP Header (Phase 2)

NPLI Phase 2 (Version S-CX7.2.0 and later) provides an extended PANI header as defined in 3GPPTS 24.229, IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3.

Extended PANI header syntax is as shown below.

```
P-Access-Network-Info  = “P-Access-Network-Info” HCOLON
                                 access-net-spec *(COMMA access-net-spec)
access-net-spec                = (access-type / access-class) *(SEMI access-info)
access-type                    = "IEEE-802.11" / "IEEE-802.11a" /
"IEEE-802.11b" / "IEEE-802.11g" /
                                 "IEEE-802.11n" / "3GPP-GERAN" / "3GPP-UTRAN-
FDD" / "3GPP-UTRAN-TDD" /
                                 "3GPP-E-UTRAN-FDD" / "3GPP-E-UTRAN-TDD" /
"ADSL" / "ADSL2" / "ADSL2+" /
                                 "RADSL" / "SDSL" / "HDSL" / "HDSL2" /
"G.SHDSL" / "VDSL" / "IDSL" /
                                 "3GPP2-1X" / "3GPP2-1X-Femto" / "3GPP2-1X-
HRPD" / "3GPP2-UMB" /
                                 "DOCSIS" / "IEEE-802.3" / "IEEE-802.3a" /
"IEEE-802.3e" / "IEEE-802.3i" /
                                 "IEEE-802.3j" / "IEEE-802.3u" / "IEEE-802.3ab"/
"IEEE-802.3ae" /
                                 "IEEE-802.3ah" / "IEEE-802.3ak" /
"IEEE-802.3aq" / "IEEE-802.3an" /
                                 "IEEE-802.3y" / "IEEE-802.3z" / GPON/ XGPON1 /
"GSTN"/ "DVB-RCS2"/ token
access-class                   = "3GPP-GERAN" / "3GPP-UTRAN" / "3GPP-E-UTRAN" /
"3GPP-WLAN" / "3GPP-GAN" /
                                 "3GPP-HSPA" / "3GPP2" / token
np                             = "network-provided"
access-info                    = cgi-3gpp / utran-cell-id-3gpp / dsl-location /
```

```
i-wlan-node-id / ci-3gpp2 /
                                  ci-3gpp2-femto / eth-location / fiber-
location / np/ gstn-location /
                                  local-time-zone / dvb-rcs2-node-id / extension-
access-info
extension-access-info          = generic-param
cgi-3gpp                       = "cgi-3gpp" EQUAL (token / quoted-string)
utran-cell-id-3gpp             = "utran-cell-id-3gpp" EQUAL (token / quoted-
string)
i-wlan-node-id                 = "i-wlan-node-id" EQUAL (token / quoted-string)
dsl-location                   = "dsl-location" EQUAL (token / quoted-string)
eth-location                   = "eth-location" EQUAL (token / quoted-string)
fiber-location                 = "fiber-location" EQUAL (token / quoted-string)
ci-3gpp2                       = "ci-3gpp2" EQUAL (token / quoted-string)
ci-3gpp2-femto                 = "ci-3gpp2-femto" EQUAL (token / quoted-string)
gstn-location                  = "gstn-location" EQUAL (token / quoted-string)
dvb-rcs2-node-id               = "dvb-rcs2-node-id" EQUAL quoted-string
local-time-zone                = "local-time-zone" EQUAL (token / quoted-string)
```

P-Subscription-MSISDN SIP Header

The Proprietary P-Subscription-MSISDN SIP header is supported in both NPLI Phases 1 (Release S-CZ7.1.2) and 2 (Release s-CZ7.2.0 and later).

Its syntax is as follows.

```
P-Subscription-MSISDN = “P-Subscription-MSISDN” HCOLON
                                  MSISDN

        MSISDN        =           msidsn value (string)
```

Handling User-Endpoint-Provided Location Information

Handling of user-endpoint-provided location information is user configurable. By default, this information is not included in the PANI header constructed by the P-CSCF. This default state can be over-ridden, in which case the PANI header will contain both user-endpoint-provided location information and NPLI.

In the case where a user-endpoint PANI header contains an **np** parameter, the location information contained in the header is ignored, and is not included in the PANI constructed by the P-CSCF.

The following sample Phase 2 PANI headers illustrate the inclusion of both user-endpoint-provided and network-provided location information.

In this case, the UE -provided location information and NPLI matched.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601; network-provided
```

In this case, the cell-IDs did not match.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207abcde7A7777; network-provided
```

In this case, neither the network access type, nor the cell type matched.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=26207FFFDF1B9601,
3GPP-GERAN; cgi-3gpp=26207FFFDF1B9601; network-provided
```

In this case, NPLI was not available.

```
P-Access-Network-Info: 3GPP-E-UTRAN-FDD; utran-cell-id-3gpp=
```

User-Endpoint-Provided Location Information Configuration

Use the following procedure to include user-endpoint-provided location information in the PANI constructed by the P-CSCF. This capability is disabled by default.

1. Move to sip-config configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

2. **include-ue-loc-info**—Use this parameter to enable the inclusion of user-endpoint-provided location information in PANI headers.

   With the parameter enabled, the PANI header contains both user-endpoint-provided location information in addition to NPLI.

   By default, **include-ue-loc-info** is disabled.

   ```
   ACMEPACKET(sip-config)# include-ue-loc-info enable
   ```

3. Use **done**, **exit** and **verify-config** to complete configuration.

NPLI Emergency Call Processing

Currently, when the P-CSCF sends AAR requests to the PCRF for an emergency call, the P-CSCF does not wait for the response (AAA) to forward the INVITE to the core. Instead, it forwards the INVITE as soon as it sends the AAR to the PCRF. Even if the subsequent AAA reports a failure response, the P-CSCF ignores the failure and allows emergency call processing.

Recent requirements issued by Bundesnetzagentur (BNetzA), the German telecom regulator, affect the processing of emergency calls originated by mobile handsets. The German requirements mandate that INVITES for establishment of emergency calls, be buffered, for some specified period of time, pending receipt of NPLI from the PCRF. If the PCRF fails to provide NPLI within that time period, the P-CSCF forwards the emergency INVITE to the network core with a PANI header populated with both the default-location-str value (if configured) and with any UE-provided location information (if available, and if configured to do so). If location information is unavailable, the P-CSCF forwards the INVITE without a PANI header.

Use the following procedure to configure a hold-timer that specifies the time (in seconds) that emergency INVITES are buffered while waiting for NPLI from the PCRF. Such a timer ensures compliance with BNetzA regulations. Users who are not subject to BNetzA authority can choose to implement a minimum timer value, which provides the network some period of time to report possibly more accurate location information.

NPLI Emergency Call Configuration

Use this procedure to buffer emergency INVITES pending the receipt of NPLI from the PCRF. This procedure is required for users subject to German telecommunications regulations. For other users, the procedure is optional.

1. Navigate to sip-config configuration mode.

```
ACMEPACKET# configure terminal
ACMEPACKET(configure)# session-router
ACMEPACKET(session-router)# sip-config
ACMEPACKET(sip-config)#
```

2. **hold-emergency-calls-for-loc-info**—Use this parameter to both enable the buffering of emergency INVITES (thus complying with German regulations), and to specify the time (in seconds) that INVITES are buffered.

   By default **hold-emergency-calls-for-loc-info** is set to the value 0, which disables the buffering of emergency INVITEs.

   Set **hold-emergency-calls-for-loc-info** to a non-zero value, within the boundaries acceptable to the German regulator, to enable the timer.

   ```
   ACMEPACKET(sip-config)# hold-emergency-calls-for-loc-info 5
   ACMEPACKET(sip-config)#
   ```

3. Use **done**, **exit** and **verify-config** to complete this configuration

Configuring Default PANI Contents

Users can configure default contents of the P-Access-Network-Info header in the event of any failure during the NPLI transfer. By default, such failure results in the generation of an empty header. In some circumstances a more informative header may be of benefit to the user.

Default PANI contents can be configured at either the SIP interface level, or at the realm level. If values are configured at both levels, the realm value takes precedence. A value configured at the sip-interface level takes precedence over a default value (empty string) specified at the realm level.

Use the following procedure to configure default PANI contents.

1. Navigate to either the sip-interface or realm configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-interface
   ACMEPACKET(sip-interface)#
   ```

   or

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# media-manager
   ACMEPACKET(media-manager)# realm-config
   ACMEPACKET(realm-config)#
   ```

2. Use the **select** command to specify the target SIP interface or realm.
3. Use the **default-location-string** parameter to specify the default contents of the P-Access-Network-Info SIP header in the event that NPLI is not available.

   ```
   ACMEPACKET(sip-interface)# default-location-sting "Location services
   temporarily unavailable."
   ACMEPACKET(sip-interface)#
   ```

   or

   ```
   ACMEPACKET(realm-config)# default-location-sting "Location services not
   unavailable."
   ACMEPACKET(realm-config)#
   ```

4. Use **done**, **exit**, and **verify-config** to complete configuration.

Caveats

Within environments that provide NPLI support:

- **reserve-incomplete** (ext-policy-srv mode) — set to enabled
- **optimize-aar** (ext-policy-srv mode) — set to disable

Rf Interface Features

Node-Functionality AVP Support

The Oracle USM sends the Node-Functionality (862) AVP in all Rf ACR messages.

The Node-Functionality AVP indicates the function that the message's source plays in the network. The CDF/CGF function that collects the ACR messages can use the information in the Node Functionality AVP for billing or analysis purposes.

In an IMS network, theOracle USM may perform the following functions: P-CSCF, E-CSCF, IBCF, BGCF (when configured as an Oracle Communications Session Router). In fact, the system might perform different roles simultaneously, so that on a call-by-call basis, the value in the Node-Functionality might change.

To accurately reflect multiple, simultaneous functions that the Oracle USM performs, the value inserted into the Node-Functionality AVP may be defined per realm. The node functionality value for a call's ACR is taken from the configuration in the ingress realm. Each realm may only be marked with a single Node Functionality value.

The system can still be configured with a single, global Node Functionality value. This is done in the SIP config's node functionality parameter. When configured, all system-generated ACRs include this value. However, if the node functionality parameter is also configured in a realm config, the ingress realm's node functionality value supersedes the global value.

The node-functionality in the **realm-config** may be configured with an empty string (). This indicates that this realm should revert to the global node-functionality value.

Node Functionality AVP Configuration

To configure a global Node Functionality AVP value:

1. Navigate to the **sip-config** configuration element.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)# sip-config
   ACMEPACKET(sip-config)#
   ```

2. Type **select** to begin configuring this object.

   ```
   ACMEPACKET(sip-config)# select
   ACMEPACKET(sip-config)#
   ```

3. **node-functionality** — Enter a global value to insert into the Node-Functionality AVP when the system sends ACRs over the Rf interface to an appropriate destination. The default is P-CSCF. Valid values are:

   - P-CSCF

- BGCF
- IBCF
- E-CSCF

4. Save and activate your configuration.

Node Functionality Per Realm Configuration

To configure a Node Functionality value per realm:

1. Navigate to the **realm-config** configuration element.

```
ACMEPACKET(session-router)# exit
ACMEPACKET(configure)# media-manager
ACMEPACKET(media-manager)# realm-config
ACMEPACKET(realm-config)#
```

2. Select the ingress realm where calls generate ACR's whose node functionality AVP will be marked as configured.

```
ACMEPACKET(realm-config)# select
identifier:
1: realm01    left-left:0      0.0.0.0
selection: 1
ACMEPACKET(realm-config)#
```

3. **node-functionality** — Enter the value to insert into the Node-Functionality AVP when the system sends ACRs for calls that enter the system from this realm. The default is empty which uses the global node functionality value configured in the sip-config configuration element. Valid values are:
   - P-CSCF
   - BGCF
   - IBCF
   - E-CSCF
4. Save and activate your configuration.

AAR Message Optimization

Currently, upon receiving an INVITE with a Proxy-Authorization header, the P-CSCF sends an Authorization-Authentication Request (AAR) message when the values of the **optimize-aar** and the **reserve-incomplete** parameters in the **ext-policy-server** configuration element are set to **enabled**. If the INVITE does not contain a Proxy-Authorization header then an AAR message is not sent. However, for mobile VoLTE, because IMS AKA is used for security there is no need to authenticate requests, so all INVITE messages are sent without Proxy-Authorization headers. This enhancement allows the P-CSCF to generate an AAR message when the INVITE is identified as coming from an IMS AKA user.

When the Oracle USM acts as a P-CSCF, the Diameter Rx interface updates bandwidth and addressing changes during a session. Many transactions between the P-CSCF and a PCRF server trigger a new SDP offer resulting in the transmission of an P-CSCF-initiated AA-Request (AAR) sent to the PCRF for the purpose of updating bearer parameters. However, not all transactions need to be reported to the PCRF, for example transactions that do no carry any bandwidth or addressing changes. In such instances, the issuance of an AAR is unnecessary and wasteful.

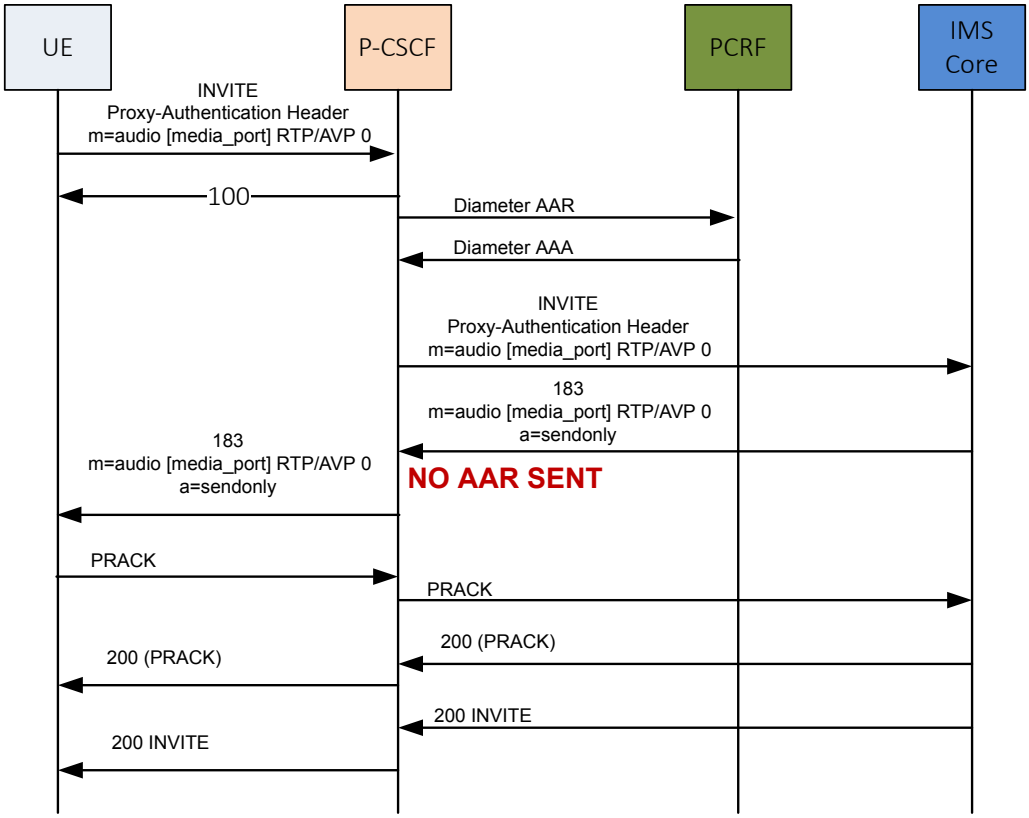
AAR message optimization (the suppression of unnecessary AARs) is activated by setting the **optimize-aar** parameter in the **ext-policy-server** configuration element to **enabled**.

If optimization is enabled, AARs are suppressed when:

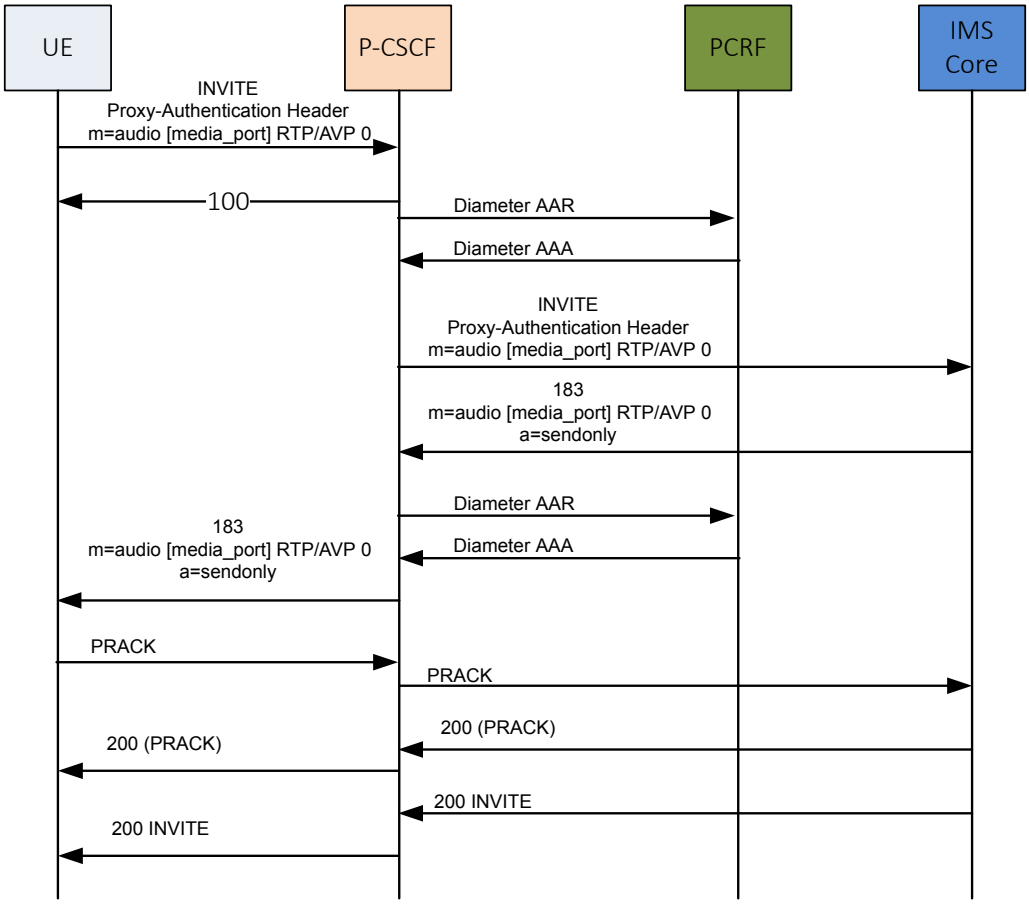
- the originating INVITE does not contain a proxy-authorization header, except for INVITE messages coming from IMS AKA users
- the codec & bandwidth provided in the m-line of an SDP response, and media IP address and port have not changed since the last SDP message
- In the MTC scenario, AAR is suppressed on reception of the originating non-IMS-AKA INVITE with no NPLI configured

AAR Optimization for Supplementary Services

In some call flows, a reINVITE or UPDATE request or response is sent to the P-CSCF and neither the bandwidth nor codec has changed. Such reINVITEs or UPDATES may include SDP offers/answers with new a=sendonly, a=recvonly, or a=inactive. These SDP parameters are directly related to the provision of Supplementary Services such as Call Hold. By default the Oracle USM does not send an AAR to the PCRF for these changes.



The Oracle USM can be configured to allow the generation of an AAR in these cases by adding the **optimized-aar=supplementary-service** option to the **ext-policy-server** configuration element. (You may also configure the option as **optimized-aar= supplementry** resulting in identical behavior.) The following call flow reflects this optimized behavior.



AAR Message Configuration

To optimize the use of AAR messages:

1. Access the **ext-policy-server** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

2. Select the **ext-policy-server** object to edit.

   ```
   ORACLE(ext-policy-server)# select
   <name>:1:  name=extpol1

   selection: 1
   ORACLE(ext-policy-server)#
   ```

3. **optimize-aar**—Set this parameter to **enabled** to optimize the use of AAR messages.
4. **reserve-incomplete**—Set this parameter to **enabled** in conjunction with **optimize-aar** set to enabled for the system to not send an AAR to the PCRF if the Proxy-authorization header is absent from the INVITE.
5. Type **done** to save your configuration.

AAR Optimization with supplementary-service Configuration

This configuration sets the Oracle USM to send AAR messages to the PCRF when in a reINVITE or UPDATEs the SDP offer/answer contains changes to the a=sendrecv, a=recvonly, a=sendonly, a=inactive lines lines.

1. Access the **ext-policy-server** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

2. Select the **ext-policy-server** object to edit.

   ```
   ORACLE(ext-policy-server)# select
   <name>:1:  name=extpol1

   selection: 1
   ORACLE(ext-policy-server)#
   ```

3. **options**—Set the options parameter by typing **options**, a space, **+optimized-aar=supplementary-service** to generate an AAR when a reINVITE or UPDATE contains SDP that changes an a=sendonly |recvonly | inactive parameter.

   ```
   ORACLE(ext-policy-server)# options +optimized-aar=supplementary-service
   ```

   > **Note:**
   >
   > You may also configure the option as **optimized-aar=supplementry** resulting in identical behavior.

   If this option is not configured and **optimize-aar** is enabled, changes to these SDP parameters will not generate an AAR.

4. Type **done** to save your configuration.

AF-Application-Identifier AVP Generation

By default, the Oracle USM populates the AF-Application-Identifier AVP with the hostname parameter of the external policy server object that receive's the AAR for the call.

The Oracle USM can populate the AF-Application-Identifier AVP with the IMS Communication Service Identifier (ICSI). 3GPP TS 24.173 defines the ICSI format as:

urn:urn-xxx:3gppservice.ims.icsi.mmtel

An ICSI appears in one of the following three SIP headers in an INVITE:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact

This AF-Application-Identifier AVP is inserted into an AAR and sent to the PCRF when the Oracle USM (as P-CSCF) receives an INVITE.

When the Oracle USM has an ICSI value from the access UE and the core, it uses the value received from the core. ICSI values received from the network supercede a configured default ICSI value. When the INVITE is received from the access side, ICSI values are taken from the access side until values are received from the core side. If no valid ICSI value is received in a message, and the Oracle USM is not configured with a default-icsi value, the external policy server object's hostname is used for the ICSI value sent to the PCRF. If an INVITE does not contain SDP, no AAR is generated.

AF-Application-Identifier AVP generation occurs for access and core initiated calls.

AVP Generation on Initial INVITE from UE

The Oracle USM receives the UE's INVITE. When configured to send an AAR on receipt of the INVITE, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact

If none of the above three headers are present, the default value of AF-Application-Identifier shall be used.

If the Oracle USM is not configured to send an AAR on receipt of an INVITE, the value from the 3 headers will be cached for later AF-Application-Identifier generation.

AVP Generation on response to Initial INVITE from UE

The Oracle USM receives the response (1xx or 200 OK) to the original INVITE. When configured to send an AAR on receipt of the response, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact header is not expected

Values obtained in this step supercede any value set from the initial INVITE step. If none of the three headers are present, the value determined in the INITIAL invite step is used for the AF-Application-Identifier AVP contents.

AVP generation on reINVITE from UE

The Oracle USM can receive a re-INVITEs from the core. The Oracle USM populates the AF-Application-Identifier AVP with one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact

If none of the three headers are present, the default AF-Application-Identifier value is used.

Responses to the Re-Invites will be treated in the similar manner.

Examples

Example 1:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P2”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 2:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP		No AAR generated
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P2”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 3:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP		No AAR generated
	200 OK with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains default AF-A-I value.
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P3”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 4:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with no SDP	AAR with AF-A-I AVP contains “P1”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P1”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 5:

Received from access UE	Received from core	Sent to PCRF
INVITE with no SDP		No AAR generated
	183 with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server’s “hostname”
PRACK with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with no SDP	No AAR generated
	200 OK (INVITE) with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P4”

Example 6:

Received from access UE	Received from core	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
	re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	AAR with AF-A-I AVP contains “P3”
200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”		AAR with AF-A-I AVP contains “P3”

AVP Generation on Initial INVITE from core

The Oracle USM receives the original INVITE. When configured to send an AAR on receipt of the INVITE, the AF-Application-Identifier AVP is populated with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service
- P-Preferred-Service
- Accept-Contact

If none of the above three headers are present, the default value of AF-Application-Identifier shall be used.

If the Oracle USM is not configured to send an AAR on receipt of an INVITE, the value from the 3 headers will be cached for later AF-Application-Identifier generation.

AVP Generation on response to initial INVITE from core

The Oracle USM receives the response (1xx or 200 OK) to the original INVITE. When configured to send an AAR on receipt of the response, if none of the three headers (P-Assured-Service, P-Preferred-Service, or Accept-Contact) were present in the original INVITE, the Oracle USM populates the AF-Application-Identifier AVP with the contents of one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact header is not expected

If none of the three headers are present, the value determined in the INITIAL invite step is used for the AF-Application-Identifier AVP contents.

AVP generation on reINVITE from core

The Oracle USM can receive re-INVITEs from the core. The Oracle USM populates the AF-Application-Identifier AVP with one of the following headers (in order of precedence) if present:

- P-Assured-Service (this header is only expected from trusted UEs)
- P-Preferred-Service
- Accept-Contact

If none of the three headers are present, the default AF-Application-Identifier value is used.

Responses to the Re-Invites will be treated in the similar manner.

Example 1:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”

Received from core	Received from access UE	Sent to PCRF
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P1”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P3”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 2:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP		No AAR generated
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P2”
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P3”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 3:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP		No AAR generated
	200 OK with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server hostname.
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P3”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 4:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with no SDP	AAR with AF-A-I AVP contains “P1”

Received from core	Received from access UE	Sent to PCRF
re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”		AAR with AF-A-I AVP contains “P3”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P3”

Example 5:

Received from core	Received from access UE	Sent to PCRF
INVITE with no SDP		No AAR generated
	183 with SDP No P-A-S, P-P-S, or A-C headers	AAR with AF-A-I AVP contains ext-policy-server’s “hostname”
PRACK with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with no SDP	No AAR generated
	200 OK (INVITE) with SDP P-A-S, P-P-S, or A-C header contains “P4”	AAR with AF-A-I AVP contains “P1”

Example 6:

Received from core	Received from access UE	Sent to PCRF
INVITE with SDP P-A-S, P-P-S, or A-C header contains “P1”		AAR with AF-A-I AVP contains “P1”
	200 OK with SDP P-A-S, P-P-S, or A-C header contains “P2”	AAR with AF-A-I AVP contains “P1”
	re-INVITE with SDP P-A-S, P-P-S, or A-C header contains “P3”	AAR with AF-A-I AVP contains “P1”
200 OK with SDP P-A-S, P-P-S, or A-C header contains “P4”		AAR with AF-A-I AVP contains “P4”

Diameter: CLF

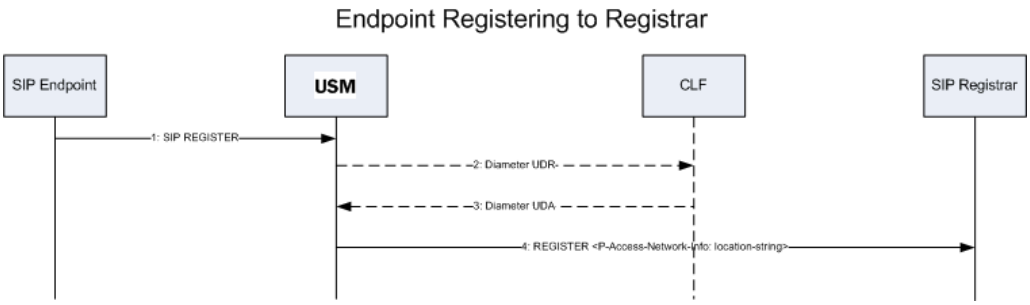
The Oracle USM supports the e2 interface over a Diameter connection acting as a P-CSCF communicating with a CLF. The application ID field must be set to 16777231 to run the e2 reference point.

A Connectivity Location Function (CLF) maintains mappings between endpoints with dynamically assigned IP addresses and their physical location. The Oracle USM, acting as a P-CSCF, is the intermediary device between a registering endpoint and a CLF. The CLF thus validates and tags a registering endpoint, and the Oracle USM applies the CLF's actions.

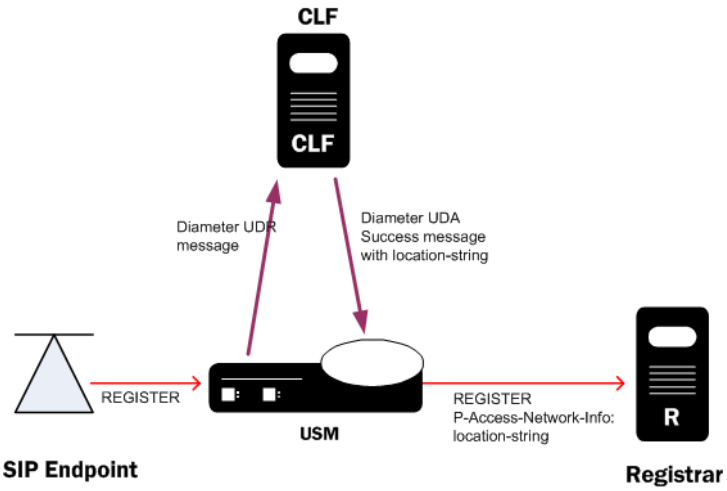
CLF Behavior

The Oracle USM and a CLF only interact with each other when an endpoint registers or re-registers. The Oracle USM, acting as the P-CSCF, is the first SIP device that the REGISTER message reaches. Upon receiving the REGISTER message(1), the Oracle USM queries the CLF using the Diameter protocol. The endpoint's (public) IP address and port, and the Oracle USM's IP information are sent to the CLF in a Diameter User-Data-Request (UDR) message(2).

The CLF responds to the Oracle USM with a Diameter User-Data-Answer (UDA) message(3). If the request is approved, then the CLF also sends a location-string value to be inserted in one of the SIP headers. The Oracle USM inserts a P-Access-Network-Info header containing the location-string into the incoming REGISTER message and forwards this message(4) to the SIP registrar/I/S-CSCF.



The Oracle USM inserts this P-Access-Network-Info header into all subsequent SIP messages from this endpoint as they are forwarded into the core network. The P-Access-Network-Info header is inserted into all SIP requests and responses except for ACK and CANCEL messages. For all boundaries where SIP messages pass from trusted to untrusted SIP interfaces or session agents, the Oracle USM will strip out the P-Access-Network-Info header as expected.



If the CLF responds with a Reject UDA message, the Oracle USM rejects the registration, and sends a 503 - Service Unavailable message back to the registering endpoint. In this way, the CLF can be used for admission control.

The Oracle USM communicates with the CLF solely for retrieving location information from the CLF, and not for notifying the CLF about an endpoint's registration state or activity. When an endpoint's registration ends, either through a normal expiration, getting rejected by the registrar, or through specific de-registering or error conditions, the Oracle USM deletes the

locally cached registration location string. The Oracle USM does not inform the CLF about any registrations that have been deleted.

P-Access-Network-Info Header Handling

The P-Access-Network-Info header is created and populated according to the following rules:

1. If the CLF returns an Accept UDA message with a location string, the Oracle USM inserts the location string into a P-Access-Network-Info header in the outgoing REGISTER message.
2. If the CLF returns an Accept UDA message without a location string, the Oracle USM inserts the configured default string into a P-Access-Network-Info header in the outgoing REGISTER message.
3. If the CLF returns an Accept UDA message without a location string and no location string is configured on Oracle USM, the outgoing REGISTER message is forwarded out of the Oracle USM, but no P-Access-Network-Info header is created for the REGISTER message.

P-CSCF PANI Enhancements

The Oracle USM always adds PANI headers to the egress SIP message with the value set to either:

- The location of information recieved from the remote CLF agent
- The default-location-string

When you enable **allow-pani-for-trusted-only** in the SIP config configuration element, the Oracle USM only forwards PANI headers to and from trusted sources or if access-info does not have network-provided in the header. A trusted domain is determined by the trust-level of the corresponding realm or SIP interface.

CLF Re-registration

The Oracle USM will send a new UDR message to the CLF to request a new location string if any of the following events occur:

1. The endpoint's contact address changes.
2. The SIP Register message's Call-ID header changes.
3. The endpoint's public IP Address or UDP port changes.
4. The endpoint connects to a different SIP interface, port, or realm on the Oracle USM than it did in the initial REGISTER message.
5. The registration expires in the Oracle USM's registration cache.

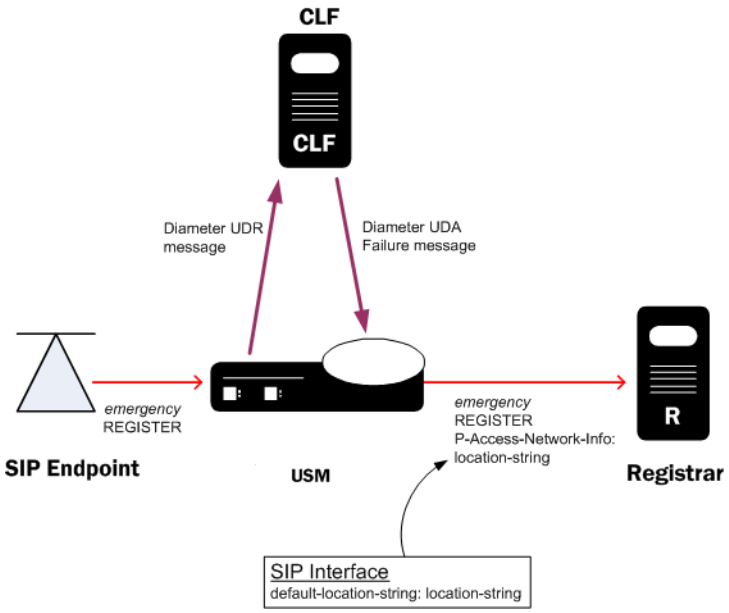
CLF Failures

If a Diameter connection fails, the Oracle USM will continually try to re-establish the connection. Endpoints that are already registered will stay registered unless they timeout or if the registrar rejects their refreshes. When the Diameter connection has not been established, and an endpoint registers on a SIP interface that is configured to use CLF, the Oracle USM forwards new REGISTER messages to the registrar using the default location string.

CLF Emergency Call Handling

The Oracle USM allows emergency calls into the network even if the endpoint that places the emergency call is not registered. In the expected fashion, the Oracle USM will query the CLF first for an incoming emergency call sourced from an unregistered endpoint. If the CLF response is successful, then the Oracle USM will insert the string returned from the CLF into a P-Access-Network-Info header, and insert this header into the emergency call's REGISTER message. If no location string is returned with a successful CLF response, the default location string is inserted into P-Access-Network-Info header.

If the CLF's response is to reject the emergency call, the Oracle USM will insert the configured default location string into the P-Access-Network-Info header and forward the emergency call's REGISTER message toward the registrar. For emergency calls where the endpoint has already successfully registered, the call will be routed into the network using the expected methods for emergency call routing.



If the Diameter connection to the CLF is down, emergency calls from un-registered endpoints are still allowed into the network using the default string inserted into the emergency messages.

CLF Diameter e2 Error Handling

When the Oracle USM receives an error, as defined in RFC 3588, in a UDA message from a CLF, it replies to the UA with a 503 Service Unavailable message. There are exceptions for certain values embedded in either the Result-Code AVP or Experimental Result-Code AVP.

Result-Code AVP

If the CLF sends a Result-Code AVP in a UDA message with a value of DIAMETER_UNABLE_TO_COMPLY (5012), the Oracle USM forwards the REGISTER message using the default configured location string in the P-Access-Network-Info header. If the configured default string is left blank the Oracle USM forwards the SIP REGISTER message without the P-Access-Network-Info header.

Experimental-Result-Code AVP

If the CLF sends a grouped Experimental Result AVP in a UDA message that is not successful, as per RFC 3588, the Oracle USM performs the following actions with respect to receipt of these two result code values:

- DIAMETER_ERROR_USER_UNKNOWN(5001): The Oracle USM inserts the configured default-string in for the P-Access-Network-Info header when forwarding the SIP message.
- DIAMETER_ USER_DATA_NOT_AVAILBLE (4100): The Oracle USM inserts the configured default-string in for the P-Access-Network-Info header when forwarding the SIP message.

For both cases, if the configured default string is left blank the Oracle USM forwards the SIP REGISTER message without the P-Access-Network-Info header. For all other failure result codes, the Register is rejected.

Diameter CLF e2 Error Bypass

When a CLF returns error messages i.e., any other Diameter answer than DIAMETER_SUCCESS, you can configure the Oracle USM to override the default behavior and still forward REGISTER messages with P-Asserted-Network-Id header containing the **default-location-string** parameter value. If the **default-location-string** parameter value is empty, the REGISTER message is forwarded without the P-Access-Network-Info header. This error bypass feature is configured by setting the **permit-on-reject** parameter to **enabled** for an external policy server configuration element with application-mode set to e2.

CLF-only AVPs

Globally-Unique-Address AVP

When endpoints registering through the Oracle USM reside in nested realms, the Oracle USM allows you to set the realm that appears in the Globally-Unique-Address AVP in Diameter UDR messages destined for a CLF.

The **ingress-realm-location** parameter in the external policy server configuration specifies whether to use the realm on which a signaling message arrived, or to use that realm's parent. If you choose to use the parent realm, the Oracle USM uses the one associated with the SIP interface on which the REGISTER request arrived.

e2 Interface

You can configure a value to be sent in the Address-Realm AVP (communicated in the Globally-Unique-Address AVP) for the Diameter e2 interface. This AVP is sent on a per-realm basis in the Location Information Query (UDR) query the Oracle USM (as a P-CSCF) sends to the CLF.

The CLF maintains details about IP connectivity access sessions associated with user equipment connected in the network. This CLF supports the standardized Diameter e2 interface with an Application Function; from it, the application function (i.e., the Oracle USM in the role of P-CSCF) retrieves location information and other data related to the access session. To do so, the AF sends a UDR containing Global-Unique-Address and Requested Information attributes.

Then the CLF returns a Location Information Response (UDA) containing either a success result code with location information or an error result code.

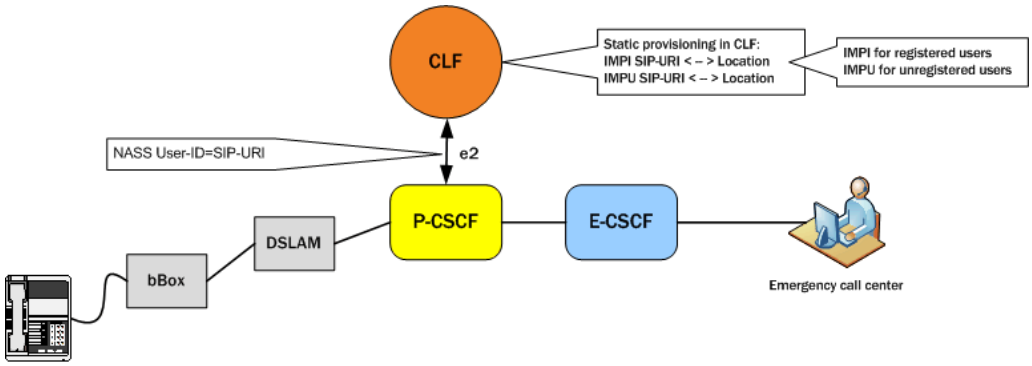
In the UDR's Global-Unique-Address, the Oracle USM currently supports the mandatory parameters: the Framed-IP-Address AVP and the Address-Realm AVP. The address realm AVP is the realm address in FQDN form, populated based on the realm on which a REGISTRATION arrived or the SIP interface. With nested realms configured, using a realm for this value can quickly become complicated.

You can configure the value sent in the Address-Realm AVP on a per-realm basis. You set the external policy server's **ingress-realm-location** parameter to the **diam-address-realm** value, pointing the Oracle USM to the associated realm from which it will learn Address-Realm AVP information. This access realm (or child realm, the realm on which enter registration came in) has its **diam-e2-address-realm** value set to the value with which to populate the Address-Realm AVP for the outgoing UDR message.

CLF e2 Interface User-Name AVP Support

In compliance with ETSI ES 283 -35 V1.2.1, the Oracle USM's e2 interface can query the CLF using the SIP endpoint's IP address or its NASS User-ID. The system's e2 interface uses the uses the SIP-URI to query the CLF by including the User-Name AVP in the User Data Request (UDR). The CLF can then furnish the P-CSCF (i.e., the Oracle USM) with the INSEE code in the Location-Information AVP in its User Data Answer (UDA).

This diagram shows how this capability works. The Oracle USM acts as the P-CSCF.



- Registering users—When it receives a registration request, the Oracle USM checks the incoming SIP interface to determine CLF use. If CLF use is unnecessary, the Oracle USM forwards the registration message to its destination.
  When CLF is required, the Oracle USM selects the AVPs to send the CLF, including the User-Name AVP before sending it a UDR. A **none** setting for this parameter means the Oracle USM does not include the User-Name AVP in any UDRs. The Oracle USM adds the User-Name AVP to the UDR if the **user-name-mode** parameter in the external policy server configuration is set to:
  - **endpoint-id**—IP address of the registering endpoint is sent as the payload for the User-Name AVP
  - **public-id**—SIP-URI portion of the TO header from the register message is sent as the payload for the User-Name AVP
  - **auth-user**—Username attribute of the Authorization header from the register is sent as the payload for the User-Name AVP; if there is no authorization header, the Oracle USM will not consult the CLF and will forward the registration message

- Unregistered users—When it receives an INVITE request, the Oracle USM checks the incoming SIP interface to determine if it should use an external policy server. If it does not need to use an external policy server, the Oracle USM forwards the INVITE message to its destination.
  When the Oracle USM does need to use an external policy server, it also checks to determine if the INVITE is in a registration and if location data in the registration cache is avaible for that endpoint. These requirements being met, the Oracle USM inserts the P-Access-Network-Info header with the location string into the INVITE it forwards to the destination. If these requirements are not met, the Oracle USM consults the CLF before forwarding the INVITE. The following describe the impact of the user-name-mode setting in such instance:
  - **endpoint-id**—IP address of the endpoint that issued the INVITE is sent as the payload for the User-Name AVP
  - **public-id and auth-user**—SIP-URI portion of the INVITE is sent as the payload for the User-Name AVP:
    With a P-Asserted-Id header present in the INVITE, the Oracle USM uses the first PAID header (if there are multiple PAID headers)

    Without a P-Asserted-Id header present in the INVITE's P-Preferred-Identity header, the Oracle USM uses the first PPI (if there are multiple PPI headers)

    With neither P-Asserted-Id nor P-Preferred-Identity header present, the Oracle USM uses the From header

HA Functionality

The location strings generated by the CLF are replicated on the standby Oracle USM in an HA pair. This is required so that a Oracle USM in an HA pair can instantly continue processing calls using the previously learned CLF information.

Configuring Diameter-based CLF

SIP Interface Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle USM in the role of an Access Oracle USM. In this example, you will configure additions to the ream configuration and the new external policy server configuration.

SIP Interface Configuration for CLF Support

To configure the SIP interface configuration for CLF support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

```
ORACLE(media-manager)# sip-interface
ORACLE(sip-interface)#
```

4. Type **select** and the number of the pre-configured sip interface you want to configure for CLF. This should be the ingress SIP interface for

   ```
   ORACLE(sip-interface)# select 1
   ORACLE(sip-interface)#
   ```

5. **ext-policy-svr**—Set this parameter to the same name as the External Policy Server configured that you configured for the CLF server.
6. **default-location-string**—Set this parameter to the default location string you want inserted into a P-Access-Network-Info header for when the CLF server does not return a unique location string.
7. Save your work using the ACLI **done** command.

External Policy Server Configuration

To configure the external policy server for use with a CLF:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

4. **name**—Set this parameter to an applicable name for this CLF instance of the external policy server. The value of this parameter will be entered in the SIP interface configuration element to reference this CLF.
5. **state**—Set this parameter to **enabled** to enable this CLF. The default value is **enabled**. The valid values are:
   - enabled | disabled
6. **operation-type**—Set this parameter to **admission-control** for the Oracle USM to communicate with a CLF. The default value is **disabled**.
7. **protocol**—Set this parameter to **DIAMETER** to connect with a CLF via the DIAMETER protocol.
8. **address**—Set this parameter to the IP address or FQDN of the CLF.
9. **port**—Set this parameter to the port which the CLF uses for Diameter transactions. Port 3868 is the default Diameter port. (The default value is **80**.) The valid range is:
   - Minimum—0
   - Maximum—65535
10. **realm**—Set this parameter to the realm where the CLF exists.
11. **permit-conn-down**—Enable or disable the Oracle USM's ability to permit calls if there is no connection to the external policy server. The default value is **disabled**. The valid values are:

- enabled | disabled

12. **permit-on-reject**—Change this parameter to enabled if you want the Oracle USM to forward the session on at a best-effort. Leave this parameter set to disabled (default), if you want the Oracle USM to deny the session on attempts to revert to the previously-requested bandwidth. (The default value is disabled.) Valid values are:

    - enabled | disabled

13. **disconnect-on-timeout—**Disable this parameter to prevent timeouts triggered by Gate-Set or Gate-Delete message sequences between the Oracle USM and a policy server from tearing down their connection. Retaining the default (enabled) allows all timeouts to tear down and re-establish the TCP connection. The valid values are:

    - **Default enabled**
    - **enabled | disabled**

14. **product-name**—Enter text string that describes the vendor-assigned name for the CLF. This parameter is required.

15. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**.

16. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the CLF. The default value is zero (**0**). For the e2 CLF reference point, the application id is **16777231**.

17. **framed-ip-addr-encoding**—Enter the format of the Frame-IP-Address (AVP 8) value in Diameter messages. The default value is **octet-string**. The valid values are:

    - **ascii-string**—Example: 192.168.10.1
    - **octet-string**—Example: 0xC0A80A01

18. **dest-realm-format**—Enter the format you want to use for the Destination-Realm AVP. The default value is **user_with_realm**. The valid values are:

    - user_with_realm | user_only | realm_only

19. **ingress-realm-location**—Set this parameter to configure the child realm or its parent for the Address-Realm in the Globally-Unique-Address AVP in Diameter UDR messages that the Oracle USM sends to the policy server. There are two choices:

    - **realm-in** (default)—This setting means that the Oracle USM will use the same realm on which the REGISTRATION request arrived.
    - **sip-interface**—This setting means that the Oracle USM will use the realm associated with the SIP interface on which the REGISTRATION request arrived.
    - **diam-address-realm**—For the e2 interface, this value enables configurable Address-Realm AVPs. This setting points the system to the associated realm from which it will learn Address-Realm AVP information. The default is realm-in.

      Remember to set the diam-e2-address-realm parameter in the realm this parameter points to.

20. **user-name-mode**—Determines how the User-Name AVP is constructed. Used primarily with e2 based CLF functionality. There are four choices.

    - none—the system does not include the User-Name AVP in any UDRs (default)
    - endpoint-ip—IP address of the registering endpoint is sent as the payload for the User-Name AVP

- public-id—SIP-URI portion of the TO header from the register message is sent as the payload for the User-Name AVP
- auth-user—Username attribute of the Authorization header from the register is sent as the payload for the User-Name AVP; if there is no authorization header, the Oracle USM will not consult the CLF and will forward the registration message

21. **domain-name-suffix**—Enter the suffix you want to use for Origin-Realm and Origin-Host AVPs that have a payload string constructed as a domain name Your value can be any string, to which the Oracle USM will prepend with a dot if you do not include one. The default value is **.com**.
22. **gate-spec-mask**—With this parameter, you can configure the Oracle USM to use a mask comprised entirely of zeros (0). The default value is 255. This parameter sets the value to use for the COPs pkt-mm-3 interface. This interface maintains a persistent TCP connection to the external policy server, even without repsonses to requests for bandwidth. This permits calls to traverse the Oracle USM even though the external policy server either fails to respond, or rejects the session.
    - Default 255
    - Values Min: 0 / Max: 255
23. **allow-srv-proxy**—Set to enabled in order to include the proxy bit in the header. The presense of the proxy bit allows the Oracle USM to tell the external policy server whether it wants the main server to handle the Diameter message, or if it is okay to proxy it to another server on the network (disabled). The default is enabled. The valid values are:
    - enabled | disabled
24. **reserve-incomplete**—Set this parameter to **enabled** if you want the Oracle USM to send a message to the CLF that does not include the endpoint's true port number. A value of 0 will be used for the port number. The default value is **enabled**. The valid values are:
    - enabled | disabled
25. **watchdog-ka-timer—**Enter the interval in seconds of Oracle USM-initiated watchdog/keep-alive messages. Valid values range between 0 and 65535 seconds.
26. **trans-expires**—Set the amount of time, in seconds, that the Oracle USM waits before performing an expiration if a Diameter base protocol transaction does not occur. The default value is 15 seconds. Valid values range between 1 and 15.
27. Save your work using the ACLI **done** command.

CLF Debugging

A new argument has been added to the show command for viewing CLF statistics. From the user prompt, type **show ext-clf-svr**.

```
ORACLE# show ext-clf-svr
14:17:14-114
EBM Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                0       0       0          0       0       0
Connections            0       0       0          0       0       0
                         ---- Lifetime ----
                    Recent      Total  PerMax
CLF Requests             0          0       0
CLF Admits               0          0       0
```

```
CLF Errors              0          0       0
CLF Rejects             0          0       0
CLF Expires             0          0       0
CLFD Errors             0          0       0
```

You can also refer to the log.ebmd log file located in the /ramdrv/logs/ directory on the Oracle USM. This file must be retrieved via FTP or SFTP.

E2 CLF Configurable Timeout

You can configure a transaction timer value for each external policy server based on the round-trip and response times for each specific policy server. Defining this transaction timer value allows you to avoid situations when policy/DIAMETER servers fail to respond to the Oracle USM 's requests, ensuring that transactions proceed in the desired amount of time and avoiding undesirable delays in set-up times.

The time you set for the DIAMETER CLF must be from 1 second to 15 seconds. For purposes of backward compatibility, 15 seconds is the default. It is recommended that you set the timer at 6 seconds or greater in accordance with the smallest DIAMETER Watchdog Request/Answer (DWR/DWA) per RFC 3588.

Activating a Configuration with the E2 CLF Timeout Defined

When you define a value for the E2 CLF timeout and activate your configuration, the change in time will only affect new client transactions. Previously existing client transactions will use the value in effect prior to your having activated the configuration.

E2 CLF Timeout Configuration

To define the E2 CLF timeout for an external policy server:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ORACLE(configure)#
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **ext-policy-server** and press Enter.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

4. **trans-expires**—Enter the time (in seconds) for the E2 CLF timeout, a transaction timer value for each external policy server based on the round-trip and response times for each specific policy server. The default for this parameter is 15 seconds; your entry must be between 1 and 15.

5. Save your work.

Diameter Policy Server High Availability

External Policy Server HA Cluster

The Oracle USM can provide external policy server redundancy through a combination of multiple servers being returned in one FQDN query and maintaining state of these servers.

When multiple IP addresses are returned in a response to a DNS query for Diameter-based external policy servers, the Oracle USM assembles the IP addresses into an HA cluster which provides redundancy for Diameter-based applications. This feature is enabled by configuring the external policy server's **address** parameter with an FQDN.

The number of servers maintained in the HA cluster is configured with the **max connections** parameter. Thus if the **max connections** parameter is set to 3, the Oracle USM maintains 1 active external policy server with 2 back up servers.

Standby Server Prioritization

The Oracle USM looks at the priority and weight fields in the DNS response to create the preferred order of the primary, secondary, tertiary, and quartiary Policy Servers These 4 addresses are known as the top-level PSs in the cluster.

The Oracle USM uses the priority and weights according to RFC 2782. However, weights only apply when multiple servers share the same priority value. If the priorities are different, then the Oracle USM will not use weights. If the priorities are the same, then weight of the two (or more) contested servers is used to determine which one to use.

Server States

An HA cluster can contain up to 4 Policy Servers, where the TCP/Diameter connection is established and monitored. Diameter session traffic is only sent to the active policy server in the cluster. The policy servers exist in the following states:

- active-TCP and Diameter connection established. Oracle USM using this server for policy decisions. The policy server with the highest priority/weight begins in this state.
- standby- TCP and Diameter connection established. Server in standby mode.
- inactive - Diameter connection not successfully established. Oracle USM tries to reconnect to inactive servers.

HA Cluster Refresh

The Oracle USM sends follow-up SRV queries to the DNS server to refresh the list of available policy servers in the cluster in the following instances:

- DNS cache expires after the TTL is exceeded
- a new policy server with FQDN for an address is configured, saved, and activated on the Oracle USM
- after an SPU switchover, the newly active SPU performs a DNS query

When the Oracle USM re-queries the DNS server for Diameter external policy servers, the cluster is refreshed with the new/changed servers. Policy server priority is also reconfigured based on newly returned priorities and weights. Upon a cluster refresh, the Oracle USM:

- closes connections with standby policy servers that are no longer in the cluster
- creates connections with policy servers which are new in the set

If the currently active policy server remains a member of the cluster after a refresh, it remains active even if its priority has changed. If the active-before-the-refresh policy server is not a member of the cluster after a DNS refresh, the Oracle USM gracefully closes the connection to this server. The Oracle USM then installs the highest priority server as the active policy server

DNS Failure

If the Oracle USM fails to get a response from the DNS server or does not receive at least one IP address in the SRV RR, it continues to send SRV queries periodically, starting with 5 seconds and doubling the interval for every sequential failure, until it receives a valid response. While waiting for a successful DNS response, the Oracle USM uses the existing Diameter servers in the DNS cache.

Policy Server Failover

The active external policy server fails over to the highest priority standby server when:

- the TCP connection is closed due to a RST or FIN
- the Diameter connection (CER/CEA exchange) could not be established
- a configured number of consecutive Diameter message timeouts occur. This number is configured in the **max timeouts** parameter. The max timeouts parameter refers to all Diameter messages except for the following:
- You can configure Diameter keepalive message time-outs separately from all other Diameter messages by setting **watchdog ka timer** in the external policy server configuration element. This will failover the active policy server based on a timeout value for DWR messages.
- You can configure Diameter STR message time-outs separately from all other Diameter messages by setting the **str-retry=**<timeout number> option in the external policy server configuration element. This will failover the active policy server based on a unique timeout value for STR messages.

If the Oracle USM sends an AAR/STR message to the active policy server and then switches to a different policy server, any new Diameter messages related to that session are sent to the same policy server as long as it is not inactive. If that server becomes inactive, messages will be sent to the new policy server. The new policy server however will not recognize the sessionID and discard the request.

External Policy Server High Availability Configuration

To configure the external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

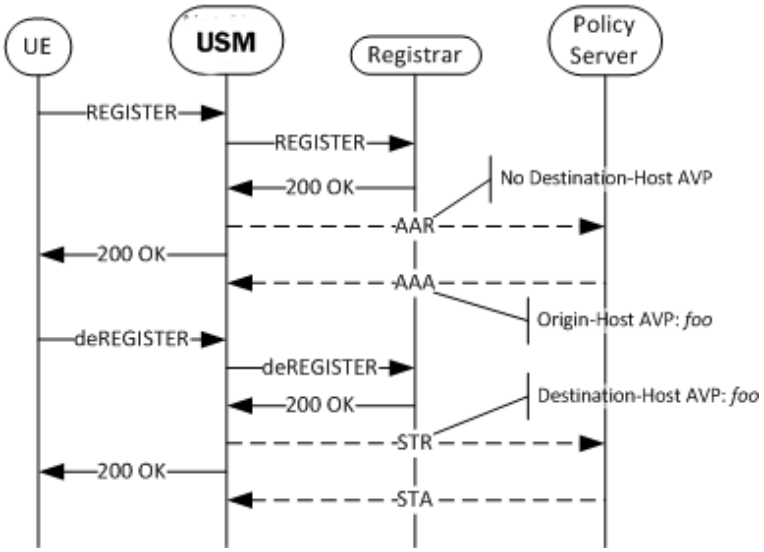
4. **address**—Enter the IP address or FQDN of the external policy server. To use external policy server redundancy, you must enter the address as an FQDN.
5. **port**—If you configure this parameter, it will override the port returned in a DNS reply.
6. **max-connections**—Set the number of servers to maintain in an external policy server cluster. Valid values range from 1 - 20.
7. **srv-selection-strategy**—Leave this parameter at its default, Failover setting.
8. **max-timeouts**—Set the number of request timeouts before the Oracle USM sets this external policy server to inactive. You can separately set the number of DWR timeouts that trigger a server to be inactive as an option. Valid values range from 0 - 200.
9. Save and activate your configuration.

Diameter Multi-tiered Policy Server Support

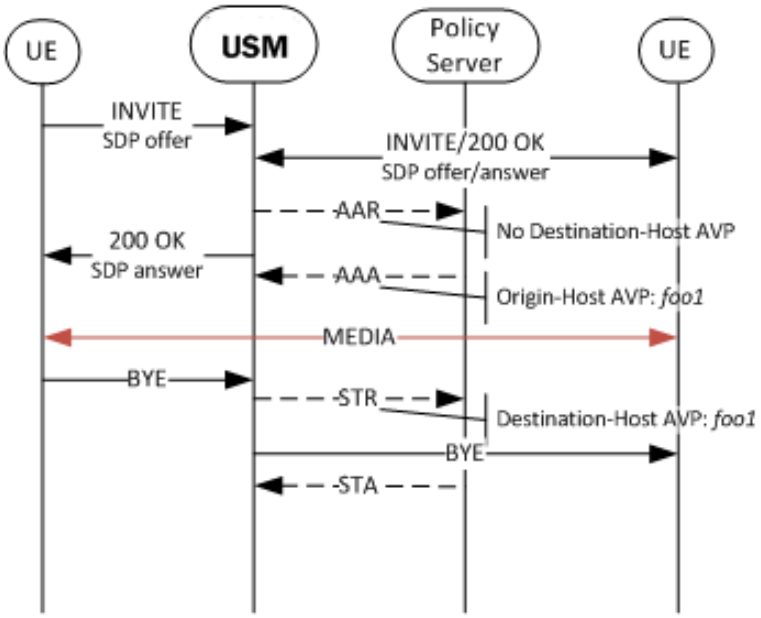
When accepting a call in a 2 tiered policy server environment, theOracle USM queries the Tier-1 policy server, running in stateless mode, for AAR-type requests. The Tier-1 PS, identifies an appropriate Tier-2 policy server for every incoming Diameter session. The mapping of session-ID to Tier-2 PS is performed between the Tier-1 and Tier-2 policy servers. This architecture is used to increase overall performance.

Two-tiered policy support is implemented per user registration and per call. To act in the P-CSCF role with a stateless Tier-1 policy server, the system recalls the Origin-Host AVP value returned in the AAA answer response to the initial Diameter AAR request. The Origin-Host-AVP value is then inserted into the Destination- Host AVP in subsequent Diameter messages (AAR, STR) corresponding to that user registration or call. Set the cache-dest-host parameter to enabled in the ext-policyserver configuration element to enable this feature.

The following call flow is an example two tiered PS support for user registration. While the initial AAR's Destination-Host AVP is empty, the system inserts the value of the received Origin-Host AVP into the registration-terminating STR.



The following call flow is an example of a two tiered PS support for a call. If this call occurs within the scope of the previous registration, the value inserted into the Destination-Host AVP will still be unique per session. Therefore, it is not the value learned from the AAA in the REGISTER, previously illustrated; it is the value learned from the AAA responding to the INVITE setting up the call.



In the case of a reINVITE changing an aspect of the call like the codec use, foo1 is inserted into the AAR's Destination-Host AVP. If the Origin-Host AVP value, as received by the system, changes mid session for a call, all subsequent Destination-Host AVP values are updated with the change.

Diameter Multi-tiered Policy Server Support

To configure Multi-tiered policy server support:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ACMEPACKET# configure terminal
   ```

2. Type media-manager and press Enter to access the media-related configurations.

   ```
   ACMEPACKET(configure)# media-manager
   ```

3. Type ext-policy-server and press Enter.

   The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ACMEPACKET(media-manager)# ext-policy-server
   ACMEPACKET(ext-policy-server)#
   ```

4. cache-dest-host—Set this parameter to enabled for the system to participate as a P-CSCF in a multi-tiered policy server environment.

5. Save and activate your configuration.

Diameter Message Manipulations

The Oracle USM can perform manipulations on all grouped and non-grouped AVPs. This is referred to as Diameter Manipulation Rules (DMR). A message manipulation is the ability to search for a predefined string within an AVP and then replace it with another value. This is similar to the Oracle USM's header manipulation rules functionality.

A diameter manipulation configuration element is defined by a name parameter. You can optionally add a description field to the diameter manipulation. Within each diameter manipulation you can configure multiple diam manipulation rule subelements. The manipulation rule subelements are the configuration where AVPs are identified, searched, and in which the data is replaced.

The user can apply diameter manipulations to external policy server configurations. These manipulations affect traffic between the Oracle USM and the applicable policy server.

> **Note:**
>
> The Oracle USM also supports diameter manipulation across the Cx interface, with the user configuring these manipulations to home subscriber server configurations. The range of manipulation supported over the Cx interface is the same as that over the Rx interface.

Manipulation Rule

Creating a manipulation rule is divided into three parts, defining the message type and AVP where the manipulation is performed, defining how the search on the AVP is performed, and defining what to replace the found string with.

You must first define the name parameter of the diam manipulation rule configuration element. Optionally you can add a descr avp code parameter that is a description of this manipulation rule.

Naming Diameter Manipulations

The Oracle USM restricts the way you can name a diameter-manipulation rule. Specifically, observe the rules below when naming manipulation elements:

- Character limit - diameter manipulation rule names cannot be longer than 24 characters.
- Numeric characters - diameter manipulation rule names must not start with a numeric character.
- Special characters - Special characters are not supported within diameter manipulation rule names, with the exception of the underscore and hyphen characters.
- Capital letter characters - The Oracle USM includes reserved keywords that are named using all-capital letters. To avoid conflicts between keywords and diameter manipulation rules, do not configure diameter manipulation rule names using ll capital letters.

Note that, although diameter-manip-rule and avp-header-rule names have the same character-type restrictions, they do not have a character limit.

Message Based Testing

When the Oracle USM first receives a message applicable for manipulation, it checks if the message type as **request**, **response**, or **all** as configured in the msg type parameter. The Oracle USM continues to look at the message command code. Matching values are defined by configuring the msg cmd code parameter with a numeric value. You can enter a single value, multiple comma-separated values, or you can leave this parameter blank to indicate all message codes.

AVP Search Value

After the Oracle USM has identified the messages where it can look for an AVP, the avp code must be defined with a numeric AVP value to be searched. Also the AVP data type is defined so Oracle USM knows how to correctly parse the AVP once found. This is configured in the avp type parameter with valid values of: octet-string, octet-hex, integer32, unsignedint32, address, utfstring, diameteruri, or enumerated.

The comparison type is defined so that the Oracle USM knows the correct way to determine if the match value appears in the avp code. Valid comparison types are:

- Case-sensitive—The comparison-type of both case-sensitive and case-insensitive literally compares the value contained in the match-value against the received value.
- Case-insensitive—The comparison-type of both case-sensitive and case-insensitive literally compares the value contained in the match-value against the received value.
- pattern-rule—the match-value is treated as a regular expression.
- boolean—Used when it is necessary to compare the results of two or several manipulation rules with varying logic (e.g. if ($rule1 & ($rule2 | $rule3))). When the comparison-type is set to boolean, the match-value will be evaluated as a boolean expression.

Finally, the match operation is configured by defining a match value, which is the string to find. The Oracle USM evaluates if the match value is found in the avp code AVP. You may also leave the match value empty for the DMR to be applied on the AVP without testing for a match.

Reserved Keywords

The Oracle USM employs certain reserved keywords as a short hand for configuration/message parameters. These keywords are as follows:

HOSTNAME—This keyword refers to the agent hostname this rule is being referenced by. If the rule is applied to a realm/interface then the value of the hostname keyword will be an empty string. If the rule is applied to the group, then the hostname for the agent picked will be used.

ORIGINREALM—This keyword refers to the Origin-Realm value for the configured realm/interface. If the rule is applied to a Diameter Director Agent, then the origin-realm value is derived from the Diameter Director Interface the agent belongs to.

ORIGINHOST—This keyword refers to the Origin-Host value for the configured realm/interface. If the rule is applied to a Diameter Director Agent, then the origin-host value is derived from the Diameter Director Interface the agent belongs to.

Actions on Found Match Value

When the match-value is found, the Oracle USM references the action parameter. This is configured as either **none**, **add**, **delete**, **replace**, **store**, **diameter-manip**, **find-replace-all**, **log** or **group-manip** and indicates the action to perform on the string. If the match-value is not found, the Oracle USM continues processing the message without any AVP manipulation. These actions mean the following:

none

None disables a manipulation rule.

add

This action inserts the value from the **new value** parameter, creates a new AVP of the specified type and adds it to the list of AVPs at the specified position.The message length and padding are re-computed to account for this newly added AVP.

delete

This action removes the specified AVP from the list of AVPs being operated on. The message length and padding will be re-computed to account for this deleted AVP.

replace

This action substitutes the existing value with the **new value** parameter. The message length and padding and AVP length and padding will be re-computed to account for changes. This is mostly applicable to variable length AVP types such as octet-string.

store

Each manipulation rule has the ability to store the data that was contained in the AVP as a string. This is useful for creating conditional logic to make decisions whether to execute other manipulation rules or to duplicate information within the Diameter message.

Every manipulation rule performs an implicit store operation prior to executing the specified action type. All store operations are based on regular expression patterns configured in the **match value**. The information that is stored in the rule is the resultant of the regular expression applied against the specified string. The **comparison-type** is ignored when the action is set to store as the Oracle USM assumes that the **match value** is a regular expression. Conditional logic cannot be used to make a decision whether to perform a store operation or not; storing always occurs. Values stored in a manipulation rule are referred to as user defined variables.

diameter-manip

When the action is set to **diameter-manip**, the Oracle USM executes the diameter-manipulation **name** configured in the **new value**. The diameter-manipulation name in the **new value** must match another diameter-manipulation name exactly (case is sensitive).

diameter-manip action type is primarily to reuse diameter-manipulations that may be common to other use cases. A diameter-manip action should never call back to itself either directly or indirectly through a different diameter-manipulation.

find-replace-all

The **find-replace-all** action searches the object's string for the regular expression defined in the match-value and replaces every matching occurrence of that expression with the value supplied in the **new value**. If the regular expression contains sub-groups, a specific sub-group can be specified to be replaced by adding the syntax [[:n:]] at the end of the expression, where n is the sub-group index (zero-based). When the action is find-replace-all, the comparison-type is ignored and the match-value is always treated as a regular expression.

group-manip

The **group manip** action is used to manipulate AVPs inside grouped AVPs. For this diameter manipulation, you must set the avp-type to **grouped**.

The **group manip** action is similar to the **diameter manip** action in that the Oracle USM executes the diameter-manipulation configured in the new value.

There is an important difference between **group manip** and diameter manip. The **diameter-manip** action is context insensitive, meaning when it jumps from one diameter-manipulation to the next diameter-manipulation, it starts looking for the specified AVP from the top of the message.

The **group manip** action is context sensitive, meaning when the processing jumps from one diameter-manipulation to the next diameter-manipulation, it will look for the specified AVP within the grouped AVP. In short, the **group manip** works at an AVP level. All actions are allowed in the subsequent manipulations that are invoked, with the key difference being that those manipulation rules will be applied to the current grouped AVP in the chain. Thus it is possible to apply manipulation to an AVP at any level in the hierarchy.

Consider the following examples:

In order to replace the experimental-result > experimental-result-code AVP value from 5002 to 3002, a **group manip** can be configured as follows:

```
diam-manipulation
        name                            manipExpRslt
        description
        diameter-manipulation-rule
                name                            expRslt
                avp-code                        297
                descr-avp-code
                avp-type                        grouped
                action                          group-manip
                comparison-type                 case-sensitive
                msg-type                        response
                msg-cmd-codes                   316,317,318
                match-value
                new-value                       expRsltCode
        last-modified-by                admin@console
        last-modified-date              2011-09-13 18:50:33
diam-manipulation
        name                            expRsltCode
        description
        diameter-manipulation-rule
                name                            expRsltCode
                avp-code                        298
                descr-avp-code
                avp-type                        unsignedint32
```

```
            action                          replace
            comparison-type                 case-sensitive
            msg-type                        response
            msg-cmd-codes                   316,317,318
            match-value                     5002
            new-value                       3002
    last-modified-by                admin@console
    last-modified-date              2011-09-13 18:56:14
```

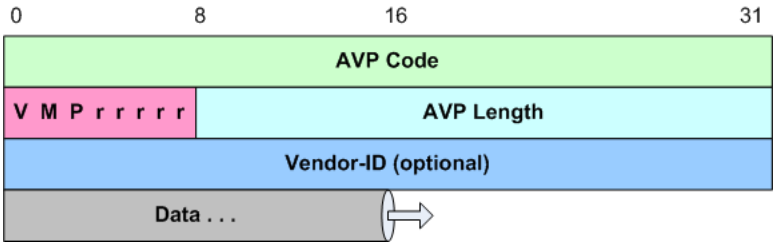
Further, if you want to add a new AVP called AvpD at the following location in the chain of AVPs Message: GrpAvpA > GrpAvpB > GrpAvpC > AvpD, then the manipulation chain would look like this

- diameter-manipulation (name=grpAvpA, action=group-manip, new-value=grpAvpB)
- diameter-manipulation (name=grpAvpB, action=group-manip, new-value=grpAvpC)
- diameter-manipulation (name=grpAvpC, action=group-manip new-value=AvpD)
- diameter-manipulation (name=AvpD action=add new-value="added new value")

> **Note:**
>
> using diameter-manip from inside the group-manip chain may have unexpected impact and must be avoided.

AVP Header Manipulation

In addition to manipulating AVPs, you can manipulate the AVP header itself. After performing AVP DMR, the AVP length and padding is recomputed. This is crucial for scenarios where a vendor-id is added or removed from the header. This functionality is configured in the avp header rules sub element. The following represents a single AVP's header:



AVP Flag Manipulation

You can manipulate the AVP flags by configuring the **header-type** parameter to **avp-flags**, this invokes operation on the flags byte in the AVP header. AVP flags are 1 byte long, where the first 3 bits represent (1) vendor, (2) must and (3) protected. The last 5 bits are reserved.

The vendor flag is critical to consider here, since it has interdependency with Vendor-Id field in the header shown above. As per RFC 3588, The 'V' bit, known as the Vendor-Specific bit, indicates whether the optional Vendor-ID field is present in the AVP header. When set, the AVP Code belongs to the specific vendor code address space. Please find specific details about the rest of the flags in RFC3588 Section 4.1.

When manipulating AVP flags, a bitwise comparison is performed between the received value and the **match value**. For ease of configuration, the **match value** is configured as a comma-separated enumerated list of vendor, must, and protected. So the **new value** and the **match value** will be used to indicate what bit in the **avp-flag** to operate on. If the **match value** is empty, the configured action is performed without any matching tests. In addition, the new value is configured using the same enumerations. The AVP header rules configuration element appears as follows:

```
avp-header-rules
   name         replaceAvpFlags
   header-type  avp-flags
   action       replace
   match-value  must,protected
   new-value         must
```

According to the example configuration, the Oracle USM makes a positive match when only the must and protected bits are set in the received avp-flags. If only the 'M' bit is set, then the match fails, the Oracle USM continues to the next header-rule.

When the match is successful (or if the **match value** is left empty), the configured **action** is performed. Consider all following actions applicable to the avp header rules sub element:

- none— no action will be performed
- add—the flags specified in the **new value** are enabled in the header, and state of the existing flags will not be changed.
  If **new value** is empty, the add operation will not be performed.

  If the **new value**=vendor, and the 'V' bit was not originally set, then the 'V' bit is now be set including a vendor-id of 9148 inserted into AVP. 9148 is the Acme Packet vendor-id assigned by IANA. It is expected that a second header-rule will be used to change this to the desired vendor-id.
- replace—all the received avp-flags will be reset. The values in the **new value** parameter will be set.
  If the **new value** is empty, the replace operation will not be performed.

  If the **new value**=vendor, and the 'V' bit was not originally set, then the 'V' bit will now be set and also a vendor-id of 9148 (Acme Packet's vendor-id) is added to the AVP header. A second header-rule may be used to change this to the desired vendor-id.

  If the **new value** does not contain vendor, and the 'V' bit was originally set, then the 'V' bit will be cleared and the vendor-id will also be set to 0 effectively removing it from the AVP header.
- delete—all flags configured in **new value**, will be deleted from the AVP header
  If the **new value** is empty, then no flags are deleted.

  If the particular flag is already empty, then it will be skipped. For example, if the **new value**=must and 'M' bit is not set, after applying the DMR the 'M' bit will still be not set.

  If the **new value**=vendor, then the 'V' bit will be cleared (if not cleared already) and the vendor-id is set to 0, effectively removing the vendor-id from the avp-header.

vendor-id Manipulation

You can manipulate the Vendor ID value in the AVP header by configuring the **header-type** parameter to **avp-vendor-id**. This performs the DMR manipulation on the 4-byte vendor-id in the AVP header. AVP vendor id is present in the AVP header only when the 'V' bit is set in the

flags. This is important because the DMR application relies upon the bit being set to determine where the data payload begins.

The avp-vendor-id search invokes an unsigned integer comparison between the received value and the **match-value**. If the **match-value** is empty, the configured action is performed without doing any match.

For the case where **match-value** is non-empty, as in the following example, the DMR engine checks whether the 'V' bit is set in the received header. If not, then the vendor id is not present either and the comparison is unsuccessful. If the 'V' bit is set, and the match succeeds, the match is successful. (An unsuccessful match has the DMR proceed to the next header-rule.)

```
avp-header-rules
   name         replaceAvpFlags
   header-type  avp-vendor-id
   action       add
   match-value  9148
   new-value    10415
```

When the match is successful (or if the **match value** is left empty), the configured **action** is performed. Consider all following actions:

- none—no action will be performed
- add—a configured vendor-id value in the **new-value** parameter is added to the AVP header and the 'V' bit set to indicate it's presence. If you prefer to set the 'V' bit in an AVP, it is better to do an avp-vendor-id action first and then manipulate the rest of the flags.
  If the **new-value** is empty, the add operation is not performed.

  If a vendor-id already exists in the AVP header, then it is replaced by **new-value**.
- replace—the existing vendor-id value is replaced with the **new-value**.
  If the **new-value** is empty, the replace operation is not performed.

  If the vendor-id does not exist in the header, then one will be added with the **new-value** and the 'V' bit is set to indicate its presence.
- delete—the vendor-id will be removed from the AVP header and 'V' bit will be reset to indicate its absence.
  If the **new-value** is empty, then the delete operation will not be performed.

  If the vendor-id does not exist, then the delete operation is not performed.

Multi-instance AVP Manipulation

Some AVPs can appear multiple times within a message. To choose a specific occurrence of an AVP, the **avp code** parameter supports indexing logic. By default, the Oracle USM operates on all instances of the specified AVP. However, after configuring an avp-code, you can specify in square brackets, a specific instance of the AVP on which to operate on. The indexing is zero-based. For example,

```
diameter-manipulation-rule
                name                          manip
                avp-code                      293[2]
```

Special characters that refer to non-discrete values are:

- Last occurrence—avp-code[^]
- All—avp-code [*]

The last (^) character is used to refer to the last occurring instance of that AVP. Any [^] refers to the first matching header that matches the specified conditional matching criteria. All [*] is the default behavior, and operates on all headers of the specified-type. For **group manip** action, the AVP index applies to the instance within that grouped AVP.

ACLI Instructions

Diameter Manipulation

To configure a diameter manipulation configuration element:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **diameter-manipulation** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# diameter-manipulation
   ORACLE(diameter-manipulation)#
   ```

4. **name**—Enter the name of this Diameter manipulation element.
5. **description**—Enter an optional description for this Diameter manipulation.
6. Type **done** and continue.

Manipulation Rule

1. Type **diameter-manip-rules** to continue and enter individual policy attributes. The system prompt changes to let you know that you can begin configuring individual parameters.
2. **name**—Enter the name of this manipulation rule. This value will be referenced in when creating a chain of rules.
3. **descr-avp-code**—Enter a description of the AVP code to manipulate.
4. **msg-cmd-code**—Enter the command code number of the message to execute the manipulation on.
5. **msg-type**—Set this to the type of message this manipulation applies to as **request**, **response**, or **all**.
6. **avp-code**—Enter the AVP by code number where this manipulation applies. You can add a multi instance identifier to the end of the avp code value, enclosed in brackets.
7. **avp-type**—Set this to the data type of the content of the match field. Refer to the Diameter standards document for the encodings of individual AVPs. Valid values are:

   none | octet-string | octet-hex | integer32 | unsignedint32 | address | diameteruri | enumerated | grouped
8. **match-value**—Enter the value within the match-field to find and make a positive match on.
9. **action**—Enter either **none, add, delete, store, diameter-manip, group-manip, find-replace-all**, or **replace** as the action to take after making a positive match on the previously entered match-value.

10. **new-value**—Enter the value that should be added or replaced in the old match-value's position.
11. Type **done** and continue.

AVP Header Manipulation

1. Type **avp-header-rules** to configure AVP header manipulation rules. The system prompt changes to let you know that you can begin configuring individual parameters.
2. **name**—Enter the name of this AVP Header manipulation rule.
3. **header-type**—Set this to either avp-flag or avp-vendor-id depending on which part of the AVP header you are manipulating.
4. **action**—Enter either none, add, delete, or replace as the action to take after making a positive match on the previously entered match-value.
5. **match-value**—Enter the value in the AVP flag field or Vendor ID field to match against.

   When matching in the avp flag field, then match-value is interpreted as comma-separated list of enumerated values <vendor,protected,must>. When matching in the Vendor ID field, then match-value is interpreted as 32 bit unsigned integer <1-4294967295>
6. **new-value**—Enter the new value when the match value is found. The resultant new value is entered as the match value is configured.
7. Type **done** to save your work.

Applying the Manipulation

You can apply a diameter manipulation by name to an external policy server configuration. This element contains the two applicable parameters: diameter-in-manip and diameter-out-manip.

> **Note:**
>
> The user can also apply diameter manipulation to a home subscriber server configuration using these same parameters.

1. To navigate to external policy server configurations from Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```
2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```
3. Enter the configuration element where you wish to apply the manipulation.
4. Type **ext-policy-server** and press Enter.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```
5. Type **select** and then choose the pre-configured external policy server you want to configure.

   ```
   ORACLE(ext-policy-server)# select
   <hostname>:
   ```

```
1: ext-pol-svr-1
2: ext-pol-svr-2
selection: 1
```

You may now add a Diameter manipulation to one or both directions of message flows.

6. **diameter-in-manip**—Enter a name of an existing diameter manipulation to apply as received by the Oracle USM on this element.
7. **diameter-out-manip**—Enter a name of an existing diameter manipulation to apply as forwarded from the Oracle USM on this element.
8. Type **done** and continue.

Diameter Manipulation Example - Supported Features AVP

This section shows you a configuration example for diameter manipulation rules. This section shows the configuration for the rule that the Oracle USM applied, and sample results of the manipulation. These examples present configurations as an entire list of fields and settings for each ruleset, nested header rules and nested element rules. If a field does not have any operation within the set, the field is shown with the setting at the default or blank.

For this manipulation rule, the Oracle USM inserts a Supported Features AVP into every request.

This is a sample of the configuration:

```
diameter-manipulation
        name                                    diamManip1
        description
        diameter-manip-rule
                name                                    rule1
                avp-code                                628
                descr-avp-code
                avp-type                                grouped
                action                                  add
                msg-type                                request
                msg-cmd-code                            265
                comparison-type                         case-sensitive
                match-value
                new-value
        diameter-manip-rule
                name                                    rule2
                avp-code                                628
                descr-avp-code
                avp-type                                grouped
                action                                  group-manip
                msg-type                                any
                msg-cmd-code
                comparison-type                         case-sensitive
                match-value
                new-value                               diamManip2
```

This second rule, which defines a new value for the first rule, builds the Feature-List-ID and Feature-List AVPs to be included within the context of the Supported Features group.

```
diameter-manipulation
        name                                    diamManip2
        description
```

```
diameter-manip-rule
        name                                    rule1
        avp-code                                266
        descr-avp-code
        avp-type                                unsignedint32
        action                                  none
        msg-type                                any
        msg-cmd-code
        comparison-type                         case-sensitive
        match-value                             10
        new-value
diameter-manip-rule
        name                                    rule2
        avp-code                                629
        descr-avp-code
        avp-type                                unsignedint32
        action                                  none
        msg-type                                any
        msg-cmd-code
        comparison-type                         case-sensitive
        match-value                             11
        new-value
diameter-manip-rule
        name                                    rule3
        avp-code                                630
        descr-avp-code
        avp-type                                unsignedint32
        action                                  none
        msg-type                                any
        msg-cmd-code
        comparison-type                         case-sensitive
        match-value                             124
        new-value
```

COPS-based External Policy Servers

The Common Open Policy Service (COPS) [RFC 2748] is a protocol supported by the Oracle USM to perform and implement Call Admission Control (CAC) based on the policies hosted in an external policy server. While the Oracle USM already supports internal CAC policies, they are not as flexible as a Resource and Admission Control Function / Policy Decision Function (RACF/PDF), the generic resource and admission control functional architecture conceived by the ITU-T and the IETF.

The Oracle USM COPS model includes a Policy server, functionally called the policy decision point (PDP), and the edge router, functionally called the policy enforcement point (PEP), the Oracle USM itself. The PDP and the PEP communicate with each other via the COPS protocol.

One of three licenses is required to use External Policy Server services: External Bandwidth Management-for RACF support.; External CLF Mgmt-for CLF support; External Policy Services-for support of both RACF and CLF.

COPS Connection

The COPS session is established over a persistent TCP connection between the PDP and PEP. A COPS Client-Open (OPN) message is sent from the Oracle USM to the RACF, which responds with a COPS Client-Accept (CAT) message. A COPS Client-Close (CC) message is sent to either side to gracefully close the persistent connection. This COPS connection is expected to never close, unless an error occurs.

COPS Failures

Connection failures are discovered through a keep alive mechanism. Keep alive (KA) messages are periodically sent by the Oracle USM to the RACF regardless if any other COPS messages have been exchanged. When a KA message is not received, a connection failure is flagged. If the COPS connection fails, the Oracle USM will continually try to re-establish the connection to the PDP. Previously established calls will continue unaffected, but the Oracle USM will deny new calls from being established until the COPS connection is restored.

Failure Detection

A COPS connection failure is triggered by one of the three following events:

1. COPS KA timeout. The Oracle USM flags a COPS KA timeout when it does not receive a response for the KA it sent to the PDP. The PDP flags a COPS KA timeout when it does not receive the KA message within its requested timer time from the Oracle USM. At a minimum, when the COPS KA message times out, the TCP socket is closed.
2. Explicit COPS CC. The Oracle USM closes a COPS connection if it receives a COPS CC message from the PDP. The PDP closes a COPS connection if it receives a CC message from the Oracle USM. After the COPS layer connection is closed, then the TCP socket is closed too.
3. TCP socket termination. If either side receives a TCP FIN or RST, the TCP socket closes as expected. The COPS layer then detects that the socket has been closed before sending any further messages, and thus the COPS connection is closed.

Failure Recovery

The Oracle USM assumes that the PDP has a mechanism that re-uses the same logical IP Address, restarts itself in a timely manner, or fails over to another PDP. Therefore, no backup PDP IP address is configured on the Oracle USM.

The Oracle USM will try to re-open the COPS connection to recover from a connection failure. The PDP is never the device to initiate a connection. The Oracle USM increases its retry interval after successive reconnect failures. Once the retry interval has grown to every five minutes, the Oracle USM continues to retry to open the COPS connection at the five minute interval.

COPS PS Connection Down

You can configure whether or not you want the Oracle USM to reject or allow new calls to be established despite the failure of a policy server (PS) connection.

You enable this feature in the external policy server configuration using a new parameter. When you enable the feature, the Oracle USM allows new SIP calls to be established even though the connection to the PS has failed. In this case, the PS will not respond and will not be aware of the established sessions. When you disable this feature, the Oracle USM behaves as it did in prior releases by responding to a connection failure with a 503 Service Unavailable.

HA Support

The Oracle USM's high availability (HA) capabilities have been extended to support COPS. When one Oracle USM in an HA configuration goes down, the MAC addresses are reassigned

to a healthy Oracle USM. IP addresses "follow" the MAC addresses to provide a seamless switchover between HA nodes.

After an HA failover, the COPS connection on the primary Oracle USM is either gracefully torn down, or times out depending on behavior of the PDP. The backup Oracle USM attempts to create a new COPS connection with the PDP. The OPN message uses the same PEPID and Client Type as in the previous pre-failover session.

Application Types

The Oracle USM supports the following COPS-based methods for interfacing with a RACF:

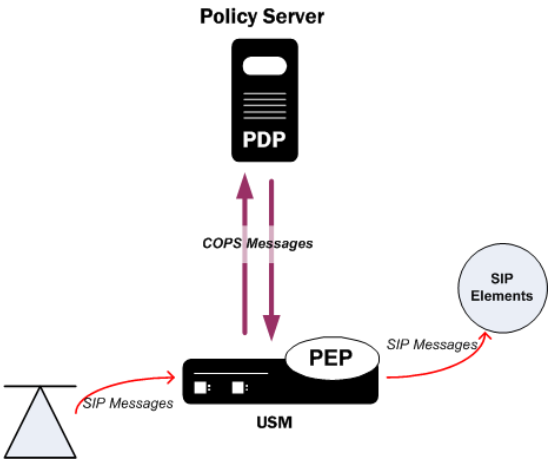
- PKT-MM3 (PacketCable™ Specification Multimedia Specification PKT-SP-MM-I03-051221) (client type: 0x800A)
- Acme Packet proprietary (client type: 0x7926)

The Oracle USM supports the following COPS-based methods for interfacing with a CLF:

- Oracle proprietary (client type: 0x7929)

COPS: RACF

CAC is performed according to the following typical scenario. When the Oracle USM receives a SIP INVITE, it sends a COPS request (REQ) message to the PDP. The REQ message includes the call ID, the SIP client's IP address, the Oracle USM's IP address and port number of the ingress interface for the call, and SDP based bandwidth requirements. The PDP responds with a COPS Decision (DEC) message with either the Install or Remove command. An Install command directs the Oracle USM to forward the INVITE to the next SIP device. A Remove command directs the Oracle USM send a SIP 503 Service Unavailable message sent back to the UA and reject the call.



The Oracle USM can be configured so that both sides of a call, based on realm, are subject to COPS bandwidth enforcement. Each flow is treated as a unique call/event, because from a media and signaling perspective, they are distinct. As the Oracle USM functions as one side of a call, its IP address is inserted into the REQ message regardless of whether it is the calling or called party. This allows for the COPS install or remove decision to be made before the Oracle USM receives the 200 OK response, and before ringing the far-end phone. Only one external policy server can be used within a single realm.

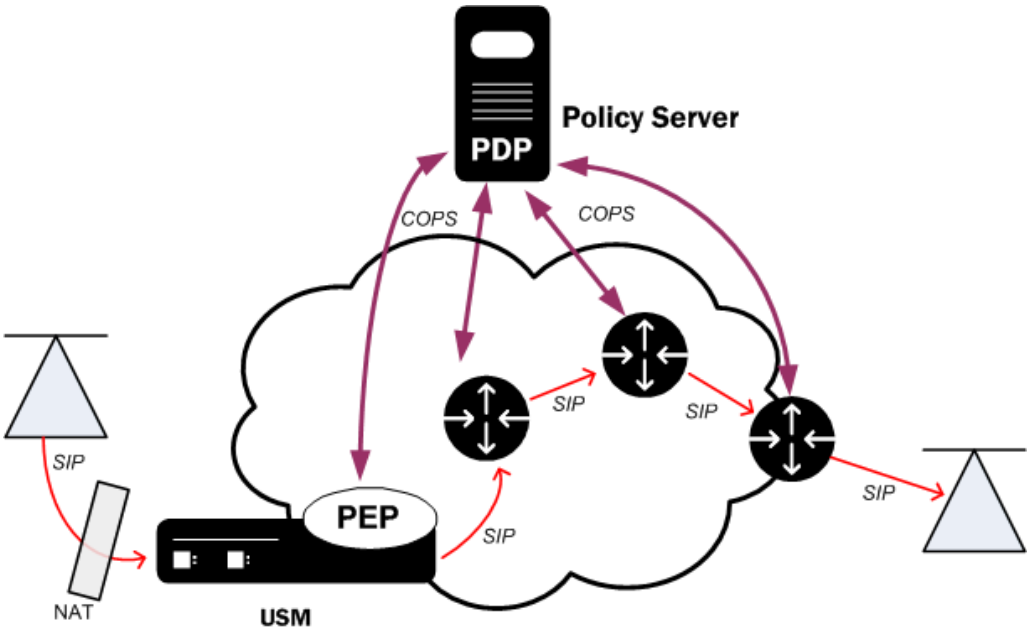
When a call ends, either with the expected SIP BYE or CANCEL methods, or due to other error conditions, the Oracle USM will delete the reservation on the PDP by sending a COPS

delete request state (DRQ) message to the PDP. All ended calls must be deleted from the PDP in order to accurately track used and available bandwidth.

Implementation Features

As the Oracle USM proxies and forwards calls, caller media information is known before the INVITE reaches the callee. The PEP can request a specific amount of bandwidth for a call, and the PDF can reserve this amount of bandwidth for a call before the called phone rings. A call's required bandwidth can also be reserved by network devices along the path from the caller to callee if the QoS admission criteria is pushed to PEPs such as routers, along this path to the callee.

The RACF can apply its hosted policies for calls originating at SIP UAs located behind NATs. This is a standard part of the Oracle USM's ability to provide seamless HNT.



Bandwidth Negotiation

Because the decision whether to admit or reject a call is made before the INVITE is forwarded to the called party, some information is not available to the PDP at the initial request. The final IP Address, UDP port number, that transport the RTP flow, and the codec used are not known by the Oracle USM until the called party responds with its own SDP information (either in the 180 or 200 response).

The Oracle USM sends a request to the PDP requesting as much bandwidth as the codec with the highest bandwidth in the SDP message requires. If the call is admitted, and when the called party returns finalized SDP information, the Session Director will modify the original reservation with the chosen codec's bandwidth requirements. This ensures the PDP has current and accurate information with which to make policy decisions.

COPS pkt-mm-3 Policy Control

You can configure the Oracle USM's COPS pkt-mm-3 interface to maintain a persistent TCP connection to the external policy server, despite there being no responses to requests for bandwidth. This permits calls to traverse the Oracle USM even though the external policy server either fails to respond or rejects the session. Without this functionality configured, the Oracle USM waits for the external policy server's authorization decision or the request to the external policy server times out, resulting in COPS pkt-mm-3 interface error responses and time-outs.

While these type of error responses and time-outs might be handled in some networks, for others they cause unacceptable call failures and call denials. To avoid this situation, you can select the granularity at which your network admits calls through a best-effort pipe and guarantee the Oracle USM's TCP connection to the external policy server remains uncompromised. The external policy server offers three parameters allowing you to enable this capability: **permit-on-reject**, **disconnect-on-timeout**, and **gate-spec-mask**.

Relationship to the TCP Connection

The TCP connection between the Oracle USM and the external policy server plays an important role because second-tier networks can experience issues between the policy server and the cable modem termination system (CMTS). Since one policy server can be connected to multiple others, one CMTS's failure can cascade and cause multiple failures for other CMTSs and policy servers.

Enables the **disconnect-on-timeout** parameter allows the Oracle USM to maintain its TCP connection to the external policy server regardless of upstream issues between PSs and CMTSs. When you disable this setting, the Oracle USM can send Gate-Set and Gate-Delete messages to in response the PS's timeouts and guard against impact to the TCP connection between the Oracle USM and the PS.

COPS Gate-Set Timeout

Without this capability enabled, the Oracle USM views the situation as a rejection when its COPS Decision Message times out at the PS. As such, the Oracle USM denies the session or attempts to revert to the previously requested bandwidth from either a G**ate**-Set add or Gate-Delete modify message. The **permit-on-reject** parameter, however, enables the system to forward the session on at a best-effort.

Enabling both the **permit-on-reject and the disconnect-on-timeout** parameters prompts yet another Oracle USM behavior. Consider the situation where a session is in the processing of being established on the Oracle USM, and it carries COPS Gate-Set messages to the PS. Typically in this scenario, one voice call has two COPS decision messages: one for the upstream audio, and one for the downstream audio. If any of the Gate-Set message do not receive a response, the Oracle USM forwards on the session under as a best effort. And if a non-final Gate-Set message in the session does not receive a response, any remaining Gate-Set messages for the session are not forwarded to the PS; the whole session is forwarded on as a best effort. In other words, if the first Oracle USM Gate-Set message out of two times out, the second for the session is not sent.

At every opportunity, the Oracle USM tried to elevate the session to a QoS session from its best-effort status. When you enable the external policy server configuration's **reserve-bandwidth** parameter, the Oracle USM checks twice for bandwidth: once upon INVITE, and a second time upon the 200 OK. Alternatively, when Oracle USM receives a ReINVITE, it

checks for bandwidth then, too. At the time of these second checks, a call's best-effort status can improve to that of a QoS session.

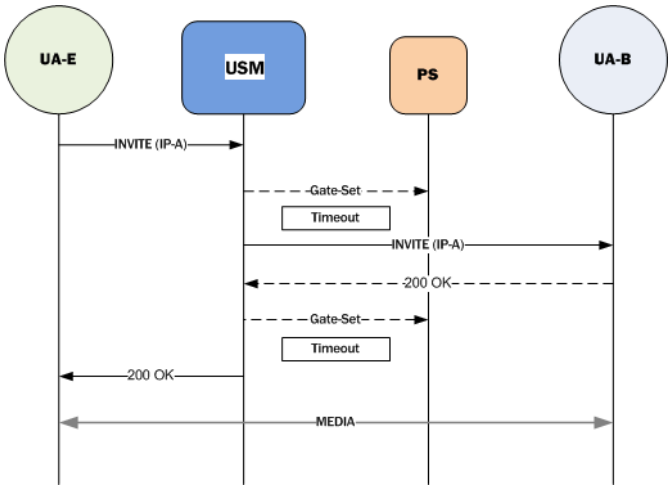
When a Gate-Set Times Out

If the Oracle USM labels a session best-effort, it does so because the G**ate**-Set message it has sent to the PS have timed out. The Oracle USM does not send a G**ate**-Delete in this situation because it has not received a valid G**ate-Id for that session from the PS.**

It is also possible that when a Gate-Set message times out, the CMTS and PS might have allocated the session after the Oracle USM's timeout. One of two errors scenarios result from this occurrence, neither of which is of significance. However, you should be aware of them:

- If a Gate-Set message times out on the INVITE side of the session, the subsequent Gate-Set message triggered by the 200 OK looks like a request to allocate the same resources. If the CMTS and PS are able to successfully install the flow outside the Oracle USM's eight-second request time-out, issues can ensue.
- Assuming the Gate-Set message time out but are nonetheless installed after the Oracle USM's request time-out, the Oracle USM does not send Gate-Delete messages for the flows because it believes they are unallocated on the PS. Because the Oracle USM has no Gate-Id data, it remains unaware the sessions are installed on the PS and CMTS. The PS and CMTS might then be left with stranded gates.

This diagram shows a call flow where the Gate-Set times out. It assumes an access-to-core configuration, where the external bandwidth management configuration shows:

- **disconn-on-timeout**=disabled
- **permit-conn-down**=enabled
- 

COPS Decision Gate-Set Message Rejected

The COPS pkt-mm-3 interface reply as it does when this feature is not enabled (reply with an error to the signaling application) when:

- The PS responds to the Gate-Set message with an error, either denying or rejecting the flow
- The **permit-on-reject** parameter is disabled (its default setting)

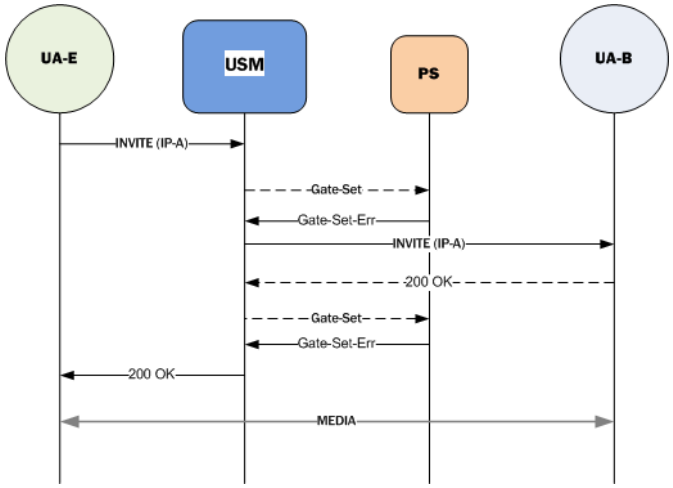
This again results in the situation where a session is in the processing of being established on the Oracle USM, and it carries COPS Gate-Set messages to the PS. Typically in this scenario, one voice call has two COPS decision messages: one for the upstream audio, and one for the downstream audio. If any of the Gate-Set message do not receive a response, the Oracle USM fowards on the session under as a best effort when the **permit-on-reject** parameter is enabled.

If a non-final Gate-Set message for a session elicits an error or deny response, the remaining Gate-Set message(s) for that session are not sent to the PS. The Oracle USM forwards the entire session as a best effort. This means that if the Oracle USM sends two Gate-Set messages and the first meets an error or deny response, then the second one is not sent. Still, the Oracle USM attempts to elevate the session's status from best-effort to QoS given the multiple requests for bandwidth it might issue.

Note

> **Note:**
>
> The Oracle USM does not send Gate-Delete message for session that have been deemed best-effort. This is because the Gate-Set messages the Oracle USM sent to the PS have timed out, and the Oracle USM does not have the PS's Gate-Id data for the session.

This diagram shows a call flow where Gate-Set messages are denied. t assumes an access-to-core configuration, where the external bandwidth management configuration shows a **permit-on-reject set to enabled.**



About the Gate-Spec Mask

Vendors of policy servers and those who incorporate PSs into their networks must allow the CMTS to overwrite IP packets with ToS bytes because the value of the Oracle USM 's DSCP/TOS mask is 0xFF. The CMTS uses the following equation to determine whether or not a ToS byte should be overwritten:

```
new-ip-tos=((orig-ip-tos AND tos-and-mask) OR (tos-or-mask)
```

Using the **gate-spec-mask** parameter, you can configure theOracle USM to use a mask comprised entirely of zeros (0s). Doing so allows the PS to reset the byte before calculations using the media policy configuration's DCSP field can take place.

COPS Debugging

A new argument has been added to the show command for viewing COPS and CAC statistics. From the user prompt, type **show ext-band-mgr**.

```
ORACLE# show ext-band-mgr
10:11:38-194
EBM Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                1       1       1          1       1       1
Connections            0       0       0          0       0       0
                         ---- Lifetime ----
                    Recent      Total  PerMax
Reserve                  0          0       0
Modify                   0          0       0
Commit                   0          0       0
Remove                   0          0       0
EBM Requests             0          0       0
EBM Installs             0          0       0
EBM Errors               0          0       0
EBM Rejects              0          0       0
EBM Expires              0          0       0
EBMD Errors              0          0       0
```

You can also refer to the log.ebmd log file located in the /ramdrv/logs/ directory on the Oracle USM . This file must be retrieved via FTP or SFTP.

COPS-based RACF Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic. In this example, you will configure additions to the ream configuration and the new external bandwidth manager configuration. You must also configure media profiles to accept bandwidth policing parameters.

Realm Configuration

To configure the realm configuration for COPS support in a CAC scenario:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure.

```
ORACLE(realm-config)# select 1
ORACLE(realm-config)#
```

5. **mm-in-realm**—Set this parameter to enabled so that calls from devices in the same realm have their media flow through the Oracle USM to be subject to COPS CAC. The default value is **disabled**. The valid values are:
   - enabled | disabled
6. **mm-in-network**—Set this parameter to **enabled** so that the Oracle USM will steer all media traveling between two endpoints located in different realms, but within the same network. If this field is set to **disabled**, then each endpoint will send its media directly to the other endpoint located in a different realm, but within the same network. The default value is **enabled**. The valid values are:
   - enabled | disabled
7. **ext-bw-manager**—Enter the name of the external bandwidth manager configuration instance to be used for external CAC for this Realm.
8. Save your work using the ACLI **done** command.

External Bandwidth Manager Configuration

To configure the external bandwidth manager:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# ext-policy-server
   ORACLE(ext-policy-server)#
   ```

4. **name**—Enter the name for this external bandwidth manager instance. This parameter is used to identify the PDP used for that will be used in each Realm configuration.
5. **state**—Set state to **enabled** to enable this external policy server.
6. **operation-type**—Enter **bandwidth-mgmt** for this external policy server configuration element to perform bandwidth management and communicate with a RACF. This sets the COPS client type to **0x7926**. If another vendor's Policy Server is supported, it will be a different protocol value.
7. **protocol**—Enter **COPS** to support COPS-based CAC. The **A-COPS** protocol implicitly sets the Oracle USM to use **0x4AC0** as the COPS client type. The default value is **C-SOAP**.
8. **address**—Enter the IP Address or FQDN of the external COPS-based policy server.
9. **port**—Enter the port number the COPS connection connects to on the PDP. The default value is **80**. (The standard port for COPS is 3288.) The valid range is:
   - Minimum—0
   - Maximum—65535

10. **realm**—Enter the name of the Realm in which this Oracle USM defines the external policy server. This is NOT necessarily the Realm that the Oracle USM performs admission requests for.
11. **application-mode**—Enter the type of interface you want to use. Your choices are: **Rq**, **Rx**, **Gq**, **e2**, and **none**. Set this to **none** or **pkt-mm3**.
12. **application-id**—Enter a numeric application ID that describes the interface used to communicate with the RACF. The default value is zero (**0**). The valid range is:
    - Minimum—0
    - Maximum—999999999
13. **permit-conn-down**—Enter enabled for the Oracle USM to establish new SIP sessions despite PS connection failure. The default value is **disabled**. The valid values are:
    - enabled | disabled
14. **reserve-incomplete**—Set this parameter to **enabled** when communicating with a PDP via COPS. The parameter allows the Oracle USM to make admission requests before learning all the details of the flows and devices (e.g., not knowing the final UDP port numbers for the RTP media streams until after the RTP has begun). The default value is **enabled**. The valid values are:
    - enabled | disabled
15. **permit-on-reject**—Set this parameter to **enabled** if you want the Oracle USM to forward the session on a best-effort basis. Leave this parameter set to **disabled**, its default, if you want the system deny the session or attempts to revert to the previously requested bandwidth.
16. **disconnect-on-timeout**—Leave this parameter set to enabled, its default, so the Oracle USM can maintain its TCP connection to the external policy server regardless of upstream issues between PSs and CMTSs. When you disable this setting, the Oracle USM can send Gate-Set and Gate-Delete messages to in response the PS's timeouts and guard against impact to the TCP connection between the Oracle USM and the PS.
17. **gate-spec-mask**—This parameter sets the value to use for Gate-Spec mask for the COPS pkt-mm-3 interface. The default is **.255**. The minimum value is **0**, and the maximum is **255**.
18. Save your work using the ACLI **done** command.

Media Profile Configuration

To configure the media profile configuration for COPS support in a CAC scenario:

Values for the following parameters can be found in the PacketCable™ Audio/Video Codecs Specification PKT-SP-CODEC-I06-050812 document.

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router path.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **media-profile** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. Type **select** and the number of the pre-configured media profile you want to configure.

```
ORACLE(media-profile)# select 1
ORACLE(media-profile)#
```

5. **peak-rate-limit**—Enter the r, P value:
   - **r**—bucket rate
   - **p**—peak rate
6. **max-burst-size**—Enter the b, m, M value:
   - **b**—Token bucket size
   - **m**—Minimum policed unit
   - **M**—Maximum datagram size
7. Save your work using the ACLI **done** command.

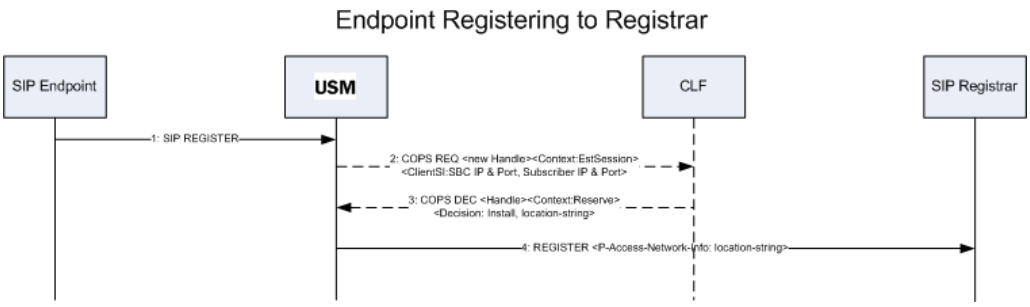
COPS: CLF

A Connectivity Location Function (CLF) maintains mappings between endpoints with dynamically assigned IP addresses and their physical location. The Oracle USM , acting as a P-CSCF, is the intermediary device between a registering endpoint and a CLF. The CLF thus validates and tags a registering endpoint, and the Oracle USM applies the CLF's actions. The Oracle USM and the CLF maintain a connection with each other using the COPS protocol.

CLF Behavior

The Oracle USM and a CLF only interact when an endpoint registers or re-registers. The Oracle USM , acting as the P-CSCF, is the first SIP device that the REGISTER message reaches. Upon receiving the REGISTER message(1), the Oracle USM queries the CLF using the COPS protocol. The endpoint's (public) IP address and port, and the Oracle USM 's IP information are sent to the CLF in a COPS REQ message(2).

The CLF responds to the Oracle USM with an Approve or Reject COPS DEC message(3). If the request is approved, then the CLF also sends a location-string value to be inserted in one of the SIP headers. The Oracle USM inserts a P-Access-Network-Info header containing the location-string into the incoming REGISTER message and forwards this message(4) to the SIP registrar/I/S-CSCF.



The Oracle USM will insert this P-Access-Network-Info header into all subsequent SIP messages from this endpoint as they are forwarded into the core network. The P-Access-Network-Info header is inserted into all SIP requests and responses except for ACK and CANCEL messages. For all boundaries where SIP messages pass from trusted to untrusted SIP interfaces or session agents, the Oracle USM will strip out the P-Access-Network-Info header as expected.

If the CLF responds with a Reject DEC message, the Oracle USM rejects the registration, and sends a 503 - Service Unavailable message back to the registering endpoint. In this way, the CLF can be used for admission control.

The Oracle USM communicates with the CLF solely for retrieving location information from the CLF, and not for notifying the CLF about an endpoint's registration state or activity. When an endpoint's registration ends, either through a normal expiration, getting rejected by the registrar, or through specific de-registering or error conditions, the Oracle USM deletes the locally cached registration location string. The Oracle USM does not update the CLF about any registrations that have been deleted.

P-Access-Network-Info Header Handling

The P-Access-Network-Info header is created and populated according to the following rules:

- If the CLF returns an Accept DEC message and a location string, the Oracle USM inserts the location string into a P-Access-Network-Info header in the outgoing REGISTER message.
- If the CLF returns an Accept DEC message without a location string, the Oracle USM inserts the configured default string into a P-Access-Network-Info header in the outgoing REGISTER message.
- If the CLF returns an Accept DEC message without a location string and no location string is configured on Oracle USM , the outgoing REGISTER message is forwarded out of the Oracle USM , but no P-Access-Network-Info header is created for the REGISTER message.

CLF Re-registration

The Oracle USM will send a new REQ message to the CLF to request a new location string if any of the following events occur:

- The endpoint's contact address changes.
- The SIP Register message's Call-ID header changes.
- The endpoint's public IP Address or UDP port changes.
- The endpoint connects to a different SIP interface, port, or realm on the Oracle USM than it did in the initial REGISTER message.
- The registration expires in the Oracle USM 's registration cache.

CLF Failures

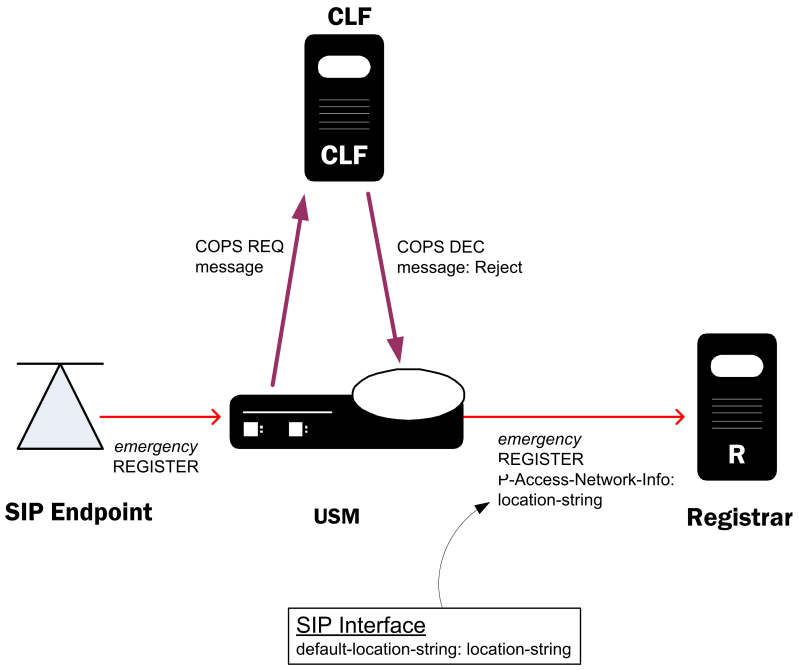
If a COPS connection fails, the Oracle USM will continually try to re-establish the connection. Endpoints that are already registered will stay registered unless they timeout or if the registrar rejects their refreshes. When the COPS connection has not been established, and an endpoint registers on a SIP interface that is configured to use CLF, the Oracle USM forwards new REGISTER messages to the registrar using the default location string.

CLF Emergency Call Handling

The Oracle USM allows emergency calls into the network even if the endpoint that places the emergency call is not registered. In the expected fashion, the Oracle USM will query the CLF first for an incoming emergency call sourced from an unregistered endpoint. If the CLF

response is successful, then the Oracle USM will insert the string returned from the CLF into a P-Access-Network-Info header, and insert this header into the emergency call's REGISTER message. If no location string is returned with a successful CLF response, the default location string is inserted into P-Access-Network-Info header.

If the CLF's response is to reject the emergency call, the Oracle USM will insert the configured default location string into the P-Access-Network-Info header and forward the emergency call's REGISTER message toward the registrar. For emergency calls where the endpoint has already successfully registered, the call will be routed into the network using the expected methods for emergency call routing.



If the COPS connection to the CLF is down, emergency calls from un-registered endpoints are still allowed into the network using the default string inserted into the emergency messages.

HA Functionality

The location strings generated by the CLF are replicated on the standby Oracle USM in an HA pair. This is required so that a Oracle USM in an HA pair can instantly continue processing calls using the previously learned CLF information.

CLF Debugging

A new argument has been added to the show command for viewing CLF statistics. From the user prompt, type **show** <space> **ext-clf-svr** <return>.

```
ORACLE# show ext-clf-svr
14:17:14-114
EBM Status                  -- Period -- -------- Lifetime --------
                   Active    High   Total      Total  PerMax    High
Client Trans            0       0       0          0       0       0
Server Trans            0       0       0          0       0       0
Sockets                 0       0       0          0       0       0
Connections             0       0       0          0       0       0
```

```
                          ---- Lifetime ----
                    Recent      Total  PerMax
CLF Requests             0          0       0
CLF Admits               0          0       0
CLF Errors               0          0       0
CLF Rejects              0          0       0
CLF Expires              0          0       0
CLFD Errors              0          0       0
```

You can also refer to the log.ebmd log file located in the /ramdrv/logs/ directory on the Oracle USM . This file must be retrieved via FTP or SFTP.

COPS-based CLF Configuration

In the following configuration examples, we assume that your baseline configuration passes SIP traffic, with the Oracle USM in the role of an Access Oracle USM . In this example, you will configure additions to the ream configuration and the new external policy server configuration.

SIP Interface Configuration

To configure the SIP interface configuration for CLF support:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the session-related configurations.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(media-manager)# sip-interface
   ORACLE(sip-interface)#
   ```

4. Type **select** and the number of the pre-configured sip interface you want to configure for CLF. This should be the ingress SIP interface for

   ```
   ORACLE(sip-interface)# select 1
   ORACLE(sip-interface)#
   ```

5. **ext-policy-svr**—Set this parameter to the same name as the External Policy Server configured that you configured for the CLF server.
6. **default-location-string**—Set this parameter to the default location string you want inserted into a P-Access-Network-Info header for when the CLF server does not return a unique location string.
7. Save your work using the ACLI **done** command.

   To configure the external policy server for use with a CLF:

8. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

9. Type **media-manager** and press Enter to access the media-related configurations.

   ```
   ORACLE(configure)# media-manager
   ```

10. Type **ext-policy-server** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

    ```
    ORACLE(media-manager)# ext-policy-server
    ORACLE(ext-policy-server)#
    ```

11. **name**—Set this parameter to an applicable name for this CLF instance of the external policy server. The value of this parameter will be entered in the SIP interface configuration element to reference this CLF.
12. **state**—Set this parameter to **enabled** to enable this CLF. The default value is **enabled**. The valid values are:
    - enabled | disabled
13. **operation-type**—Set this parameter to **admission-control** for the Oracle USM to communicate with a CLF. The default value is **disabled**.
14. **protocol**—Set this parameter to **COPS** to connect with a CLF via the COPS protocol. The default value is **C-SOAP**. The valid values are:
    - **COPS**—Standard COPS implementation. COPS client type is 0x7929 for CLF, and 0x7926 for PDP/RACF usage as defined in the operation-type parameter.
    - **A-COPS**—Vendor specific protocol. COPS client type is 0x4AC0 for admission-control operation-type.
15. **address**—Set this parameter to the IP address or FQDN of the CLF.
16. **port**—Set this parameter to the port which the CLF uses for COPS transactions. The standard port for COPS is **3288**. The default value is **80**. The valid range is:
    - Minimum—0
    - Maximum—65535
17. **realm**—Set this parameter to the realm in which the CLF exists.
18. **num-connections**—Set this parameter to the number of connections the Oracle USM will create with the CLF. The default value is **1**. The valid range is:
    - Minimum—0
    - Maximum—65535
19. **reserve-incomplete**—Set this parameter to **enabled** if you want the Oracle USM to send a COPS REQ message to the CLF that does not include the endpoint's true port number. A value of 0 will be used for the port number. The default value is **enabled**. The valid values are:
    - enabled | disabled
20. Save your work using the ACLI **done** command.

Application Type / Interface Matrix Reference

Bandwidth Management Applications

Standards Reference Point	SBC as	External Policy Server as	Protocol	Interface
TISPAN	P-CSCF	A-RACF	Diameter	Rq

Standards Reference Point	SBC as	External Policy Server as	Protocol	Interface
3GPP (R7)	P-CSCF	PCRF	Diameter	Rx
3GPP (R6)	P-CSCF (AF)	PDF	Diameter	Gq
Packet Cable	P-CSCF (AF)	PDF	COPS	PKT-MM3
		RACF	ACME COPS	

Emergency Location Services

Standards Reference Point	SBC as	External Policy Server as	Protocol	Interface
	P-CSCF	CLF	Diameter	e2
		CLF	ACME COPS	

19
Transcoding

Introduction

Transcoding is the ability to convert between media streams that are based upon disparate codecs. The Oracle USM supports IP-to-IP transcoding for SIP sessions, and can connect two voice streams that use different coding algorithms with one another.

This ability allows providers to:

- Handle the complexity of network connections and the range of media codecs with great flexibility
- Optimize bandwidth availability by enforcing the use of different compression codecs
- Normalize traffic in the core network to a single codec
- Enact interconnection agreements between peer VoIP networks to use approved codecs

By providing transcoding capabilities at the network edge rather than employing core network resources for the same functions, the Oracle USM provides cost savings. It also provides a greater degree of flexibility and control over the codec(s) used in providers' networks and the network with which they interconnect.

In addition, placing the transcoding function in the Oracle USM and at the network edge means that transcoding can be performed on the ingress and egress of the network. The Oracle USM transcodes media flows between networks that use incompatible codecs, and avoids back-hauling traffic to a centralized location, alleviating the need for multimedia resource function processors (MRFPs) and media gateways (MGWs) to support large numbers of codecs. This maximizes channel density usage for the MRFPs and MGWs so that they can reserve them for their own specialized functions.

Transcoding Hardware

A transcoding Network Interface Unit (NIU) provides the DSP resources that enable transcoding on the Oracle USM.

- The Acme Packet 6300 and Acme Packet 4600 can accept 1-24 transcoding modules to provide increasing transcoding capacity.
- The Acme Packet 3820 and the Acme Packet 4500 can accept 1-12 transcoding modules to provide increasing transcoding capacity.

Transcoding Capacity

Transcoding capacity depends on the following:

- Codecs used for transcoding
- Number of transcoding modules installed in the system. Capacity scales linearly with each extra transcoding module installed.

Transcodable Codec Details

The following table lists the supported codecs, bit rates, RTP payload type, default ptime, and supported ptimes.

Codec	Supported Bit Rate (kbps)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
G.711 PCMU	64	0	20	10, 20, 30, 40, 50, 60
G.711 PCMA	64	8	20	10, 20, 30, 40, 50, 60
G.722	48, 56, 64	9	20	10, 20, 30, 40
G.723.1	5.3, 6.3	4	30	30, 60, 90
G.726	16, 24, 32, 40	2, 96-127	20	10, 20, 30, 40, 50
iLBC	13.33	96-127	20	20, 30, 40, 60
	15.2	96-127	30	20, 30, 40, 60
G.729/A/B	8	18	20	10, 20, 30, 40, 50, 60, 70, 80, 90
AMR	4.75, 5.15, 5.90, 6.70, 7.40, 7.95, 10.2, 12.2	96-127	20	20, 40, 60, 80, 100
AMR-WB (G. 722.2)	6.6, 8.85, 12.65, 14.25, 15.85, 18.25, 19.85, 23.05, 23.85	96-127	20	20, 40, 60, 80, 100
GSM FR	13	3	20	20
T.38	4.8, 9.6, 14.4	N/A		10, 20, 30

The following table lists the supported codecs, bit rates, RTP payload number, default ptime, and supported ptimes for codecs available only on the Oracle USM 4600 and 6300.

Codec	Supported Bit Rate (kbps)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
EVRC	0.8, 4.0, 8.55	96-127	20	20, 40, 60
EVRC0	0.8, 4.0, 8.55	96-127	20	20
EVRC1	4.0, 8.55	96-127	20	20, 40, 60, 80, 100
EVRCB	0.8, 2.0, 4.0, 8.55	96-127	20	20
EVRCB0	0.8, 2.0, 4.0, 8.55	96-127	20	20
EVRCB1	4.0, 8.55	96-127	20	20, 40, 60, 80, 100
Opus	48	104	20	10, 20, 40, 60, 80, 100
SILK	8, 16	103	20	20, 40, 60, 80, 100

T.38 FAX Support

This release supports T.38 FAX relay (Version 0) conversion to T.30 over G.711 and supports FAX modulation schemes up to 14400 kbps V.17. The initial release does not support V.34 modulation.

Session Licensing

AMR/AMR-WB and EVRC/EVRC-B audio codecs require extra licenses while all other codecs are implicitly licensed by installation of a transcoding NIU. These codecs are under royalty agreements which necessitates their own special licenses.

Licenses for AMR/AMR-WB and EVRC/EVRC-B transcoding sessions are installed by groups of 25.

Transcoding Configuration

The Oracle USM performs transcoding functions— allowing the entities with incompatible codecs to communicate with each other— between two call legs. The two endpoints can be located in one or two realms or networks. The Oracle USM decides to transcode a call by evaluating messages in the SDP offer-answer transaction with respect to system configuration. An SDP offer can be in a SIP message such as an INVITE or a reINVITE, and contains information about the codecs the offerer would like to use. The answerer answers the SDP offer with its own set of supported codecs. reINVITEs are treated as new negotiations, with respect to the actual SDP offerer and answerer. The Oracle USM can manipulate an SDP message by reordering codec preference, and by adding and deleting codecs.

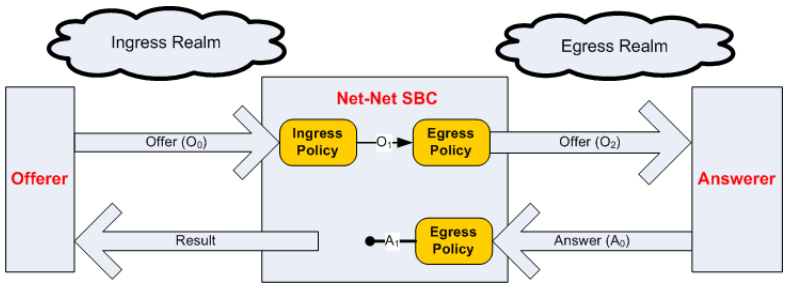
Transcoding Processing Overview

Transcoding processing is viewed in terms of the ingress and egress realms. The ingress realm is where the SDP offer is received by the Oracle USM. The egress realm is where the SDP offer is sent, and where the SDP answer is expected to be received from (i.e., the answerer's realm). A call is defined as transcodable if an egress or ingress policy exists for the session and if the session is not subject to media release, as specified in the realm configuration.

To understand the details of transcoding processing, refer to the following diagram. An SDP offer, O0, is received by the Oracle USM in the ingress realm. The ingress codec policy is applied here and the SDP offer is transformed into O1. O1 is then passed to and processed by the egress codec policy. This SDP message is then forwarded to the answerer as O2. The answerer replies with A0 to the Oracle USM, which is subjected to the egress codec policy again and transformed into A1.

When policy dictates not to transcode the call, the Result SDP sent back to the offerer is based on the common codecs shared between A1 and O1. The Oracle USM first constructs the list of codecs that are present in both in O1 and A1. Then, the Oracle USM maintains the codec order from A1 for the Result as it is sent to the offerer.

When policy dictates to transcode the call, the top transcodable codec in O1 is used in the ingress realm and the top non-signaling codec in A1 is used in the egress realm.



Defining Codec Policies

The following definitions are required for understanding transcoding processing:

DTMFable Codecs—Uncompressed codecs that are capable of properly transmitting a DTMF waveform. The only codecs designated as DTMFable are PCMU and PCMA.

FAXable Codecs—Uncompressed codecs that are capable of properly transmitting a T.30 waveform. The only codecs designated as FAXable are PCMU and PCMA.

Signaling Codecs—Non-audio codecs that are interleaved into a media stream but cannot be used on their own. The only codecs designated as Signaling Codecs by the Oracle USM are telephone-event and CN (comfort noise).

Disabling an m= line—This is in reference to an m= line in an SDP message. It means setting the m= line's port to 0 (RFC 3264). Enabling an m= line means it has a non-zero port. The m= line's mode attribute (sendrecv/inactive/rtcponly, etc) is not considered.

A codec policy is created by configuring the following information:

- Which Codecs are allowed and which are denied in a realm.
- Which Codecs should be added to the SDP m= lines for an egress realm.
- The preferred order of codecs to indicate in an SDP m= line.
- The packetization time which should be enforced within a realm.

Ingress Policy

Incoming SDP is first subject to the ingress codec policy. If no codec policy is specified in the realm config for the ingress realm, or the m= lines in the SDP offer are disabled (by a 0 port number), the SDP is transformed to O1 unchanged.

The ingress codec policy first removes all un-allowed codecs, as configured in the allow-codecs parameter by setting their port to 0 or removing the codecs from a shared m-line. For example, if two codecs share an m-line and one of them is un-allowed, the resulting m-line will not include the un-allowed codec and its attribute lines will be removed. If a single codec is used, the resulting m-line will include the codec, but its port will be set to 0 and its attribute lines will remain. Next, the remaining codecs are ordered with the **order-codecs** parameter. Ordering is when the codec policy rearranges the codecs in the SDP m= line. This is useful to suggest the codec preferences to impose within the egress realm. O1 is then processed by the egress codec policy after a realm is chosen as the destination.

In practical terms, the ingress policy can be used for filtering high-bandwidth codecs from the access realm. It can also be used for creating a suggested, prioritized list of codecs to use in the ingress realm.

Egress Policy

The Oracle USM applies egress codec policy to the SDP that has already been processed by ingress policy. The egress policy is applied before the SDP exits the system into the egress realm. If no egress codec policy is defined, or the SDP's m= lines are disabled (with a 0 port), the SDP is passed untouched from the ingress policy into the egress network.

The egress codec policy first removes all un-allowed codecs in the **allow-codecs** parameter (<codec>:no). Codecs on the **add-codecs-on-egress** list are not removed from the egress policy regardless of the how the **allow-codecs** parameter is configured. If the result does not contain

any non-signaling codecs, the ptime attribute is removed from the SDP. Codecs not present in O1 that are configured in the **add-codecs-on-egress** parameter are added to the SDP, only if O1 contains one or more transcodable codecs.

> **Note:**
>
> Transcoding can only occur for a call if you have configured the add-codecs-on-egress parameter in an egress codec policy.

If codecs with dynamic payload types (those between 96 and 127, inclusive) are added to the SDP, the lowest unused payload number in this range is used for the added codec.

The following rules are also applied for egress policy processing:

If O1 contains at least one transcodable codec, the codecs listed in the Egress policy are added to the SDP.

- telephone-event, as configured in add-codecs-on-egress will only be added if O1contains at least one DTMFable codec.
- T.38, as configured in add-codecs-on-egress will only be added if there is no T.38 and there is at least one FAXable codec (G711Fall Back (FB)) in O1. T.38 will then be added as a new m= image line to the end of SDP.
  If G711FB is not allowed in the egress policy, the Oracle USM disables the m= line with the FAXable codec. Otherwise if G711FB is allowed, pass it through the regular offer processing allowing/adding only FAXable codecs.
- G711FB, as configured in add-codecs-on-egress will only be added if there is no G711FB and there is T.38 in O1. G711FB will then be added as a new m= audio line to the end of SDP.
  If T.38 is not allowed in the egress policy, the Oracle USM disables the m= image line. Otherwise if T.38 is allowed, pass it through the regular offer processing.

If the result of the egress policy does not contain any non-signaling codecs, audio or video, the m= line is disabled, by setting the port number to 0.

The m= line is next ordered according to rules for the order-codecs parameter.

Finally, all attributes, a= lines, ptime attribute, and all other unrecognized attributes are maintained from O1. Likewise, appropriate attributes for codecs added by the add on egress parameter are added to SDP. Finally, rtpmap and fmtp parameters are retained for codecs not removed from the original offer. The result of all this is O2, as shown in the overview diagram.

In practical terms, codec policies can be used to normalize codecs and ptime in the core realm where the network conditions are clearly defined.

Codec policies can also be used to force the most bandwidth-conserving codecs anywhere in the network.

Post Processing

If any errors are encountered during the Ingress and Egress policy application, or other violations of RFC3264 occur, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of the initial call setup, the call is rejected. If O2 does not contain any enabled m= lines at the conclusion of a reINVITE, the reINVITE is rejected and the call reverts back to its previous state.

Codec Policy Definition

Codec policies describe how to manipulate SDP messages as they cross the Oracle USM. The Oracle USM bases its decision to transcode a call on codec policy configuration and the SDP. Each codec policy specifies a set of rules to be used for determining what codecs are retained, removed, and how they are ordered within SDP.

Syntax

The following parameters are used to create a codec policy. Their syntax is described inline.

allow-codecs

allow-codecs—The allow-codecs parameter configures the codecs that are allowed and/or removed from the SDP. A blank list allows nothing, ***** allows all codecs, **none** removes all codecs, the **:no** designation blocks the specific codec or class of media, and the **:force** designation is used to remove all non-forced codecs.

The **allow-codecs** parameter is configured in the following way:

- <codec>**:no**—blocks the specific codec
- *****—allow all codecs.
- <codec>**:force—**If any forced codec is present in an SDP offer, all non-forced codecs are stripped from the m- line.
- **audio:no**—audio m= line is disabled
- **video:no**—video m= line is disabled

For example, if you configure PCMU in the allow-codecs parameter, the PCMU codec, received in an SDP message is allowed on to the next step of transcoding processing, and all other codecs are removed.

The order of precedence is for removing codecs according to codec policy is:

1. <codec>**:no**—Overrides all other allow-codecs parameter actions.
2. **audio:no** / **video:no**. An allow-codecs line like “allow-codecs PCMU audio:no” disables the PCMU m= line because audio:no has a higher precedence than the specific codec.
3. <codec>**:force**
4. <codec> Specific codec name and those codecs configured in the add-codecs-on-egress list.
5. ***** has the lowest precedence of all flags. For example "**allow-codecs * PCMU:no**" allows all codecs except PCMU.

order-codecs

order-codecs—The order-codecs parameter is used to re-order the codecs in the m= line as the SDP is passed on to the next step. This parameter overwrites the order modified by the **add-codecs-on-egress** command, when relevant. The following is valid syntax for this parameter:

- <blank>—Do not re-order codecs

- *—You can add a <codec> before or after the * which means to place all unnamed codecs before or after (the position of the *) the named codec. For example:
- <codec> *—Puts the named codec at the front of the codec list.
- * <codec>—Puts the named codec at the end of the codec list.
- <codec1 > * <codec2>—Puts <codec1> first, <codec2> last, and all other unspecified codecs in between them.
- <codec>—When the * is not specified, it is assumed to be at the end.

Any codec name is allowed in the **order-codecs** parameter, even those not defined or not transcodable. An * tells the order-codecs parameter where to place unspecified codecs with respect to the named codecs. Refer to the examples below.

- <blank>—do not reorder m= line
- PCMU *—Place PCMU codec first, all others follow
- * PCMU—Place PCMU codec last, all others proceed PCMU
- G729 * PCMU—Place G729 codec first, PCMU codec last, all others remain in between
- PCMU—If * is not specified, it is assumed to be at the (PCMU *).

Add on Egress

add-codecs-on-egress—This parameter adds a codec to the SDP's m= line only when the codec policy is referenced from an egress realm (except in one 2833 scenario). Codecs entered for this parameter are added to the front of the m= line. Signaling codecs are added to the end of the m= line.

Transcoding can only occur if this parameter is configured. There is a special case for 2833 support where the add-codecs-on-egress parameter is configured for an ingress realm. See RFC 2833 Scenario 2 for details.

Packetization Time

packetization-time—This parameter specifies a media packetization time in ms to use within the realm referencing this codec policy. Packetization time You must also enable the force ptime parameter to enable transrating in conjunction with configuring the packetization time. See the Transrating section for more information.

Answer Processing and Examples

Unoffered Codec Reordering

According to RFC 3264, the answerer can add codecs that were not offered to the Answer. The answerer may add new codecs as a means of advertising capabilities. RFC 3264 stipulates that these unoffered codecs must not be used. The RFC does not dictate where in the m= line these codecs can appear and it is valid that they may appear as the most preferred codecs.

In order to simplify the answer processing, the Oracle USM moves all unoffered codecs in A0 to the back of the SDP answer before any other answer processing is applied.

Non-transcoded Call

The decision to transcode is based on the top non-signaling codec in A1. If the top A1 codec is present in O1, the call proceeds, non-transcoded. This is the rule for non-signaling codecs (i.e., not RFC 2833 nor FAX).

Transcoded Call

The following two conditions must then be met to transcode the call's non-signaling media:

- The top A1 codec is not present in the O1 m= line
- The top A1 codec was added by the egress policy

If these rule are met, the Oracle USM will transcode between the top A1 codec and the top transcodable, non-signaling O1 codec.

Voice Transcoding

The following examples use the ingress and egress codec policies listed at the top of each scenario. The examples use changing SDP offers and answers, which contribute to unique results, per example. The effects of the SDP offers and answers are explained in each example.

Voice Scenario 1

The following ingress and egress policies are used for scenario 1.

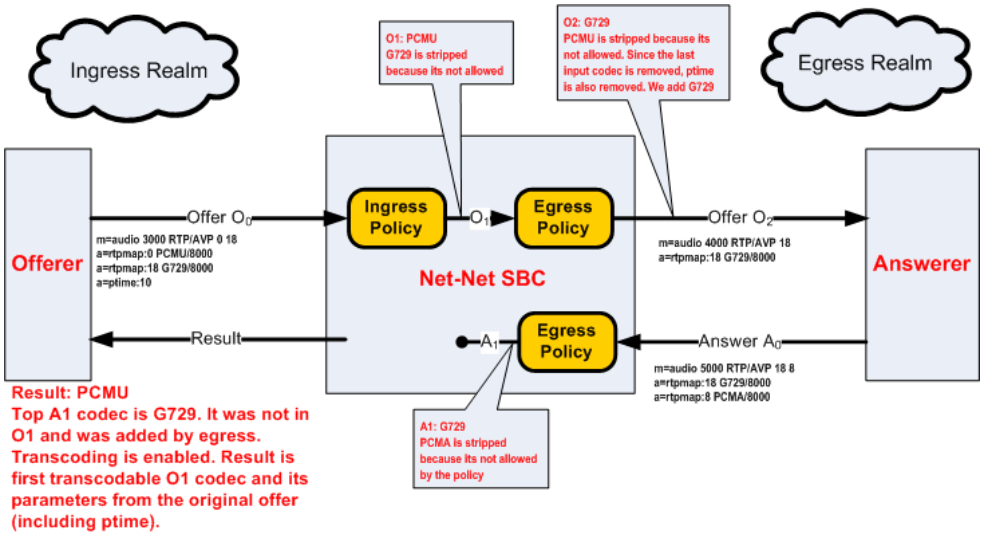
Ingress Policy		Egress Policy	
allow-codecs	PCMU GSM	allow-codecs	G729 GSM G722
add-codecs-on-egress	PCMU	add-codecs-on-egress	G729
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

Note:

The codec in the ingress policy's add-codecs-on-egress parameter has no effect in the following examples. Its presence would have an effect if a reINVITE was initiated from egress realm, effectively reversing the roles of the codec policies.

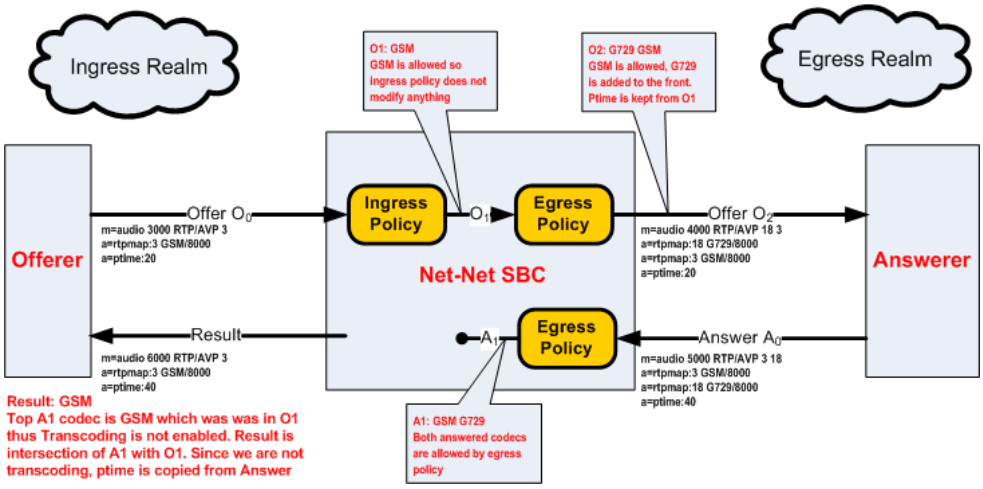
1. In the following diagram, PCMU and G729 are offered. Ingress policy removes G729 and allows PCMU. The egress policy adds G729 and strips PCMU from offered SDP and forwards it on to the answerer (ptime is also removed because the last codec was removed).

   The SDP answer agreed to use G729 and adds PCMA. The egress policy then strips PCMA from the SDP answer. At this point, the top codec in A1, G729 is checked against O1. Since G729 is not present in O1, it is transcoded to PCMU.

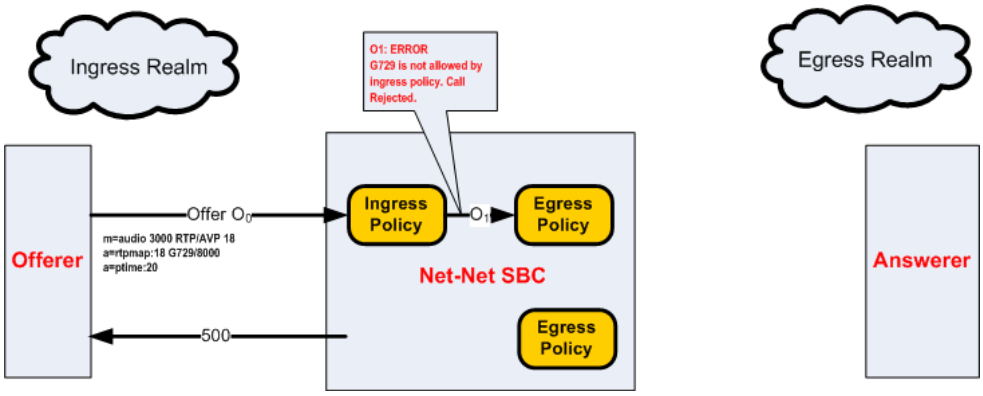


2. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from GSM and sends this to the answerer as O2.

   The SDP answer agrees to use G729 and GSM, but prioritizes GSM. The egress policy allows both codecs through, unchanged. Because A1 and O1 both have GSM, it is used for the non-transcoded call. Ptime is copied from A0 to the result.

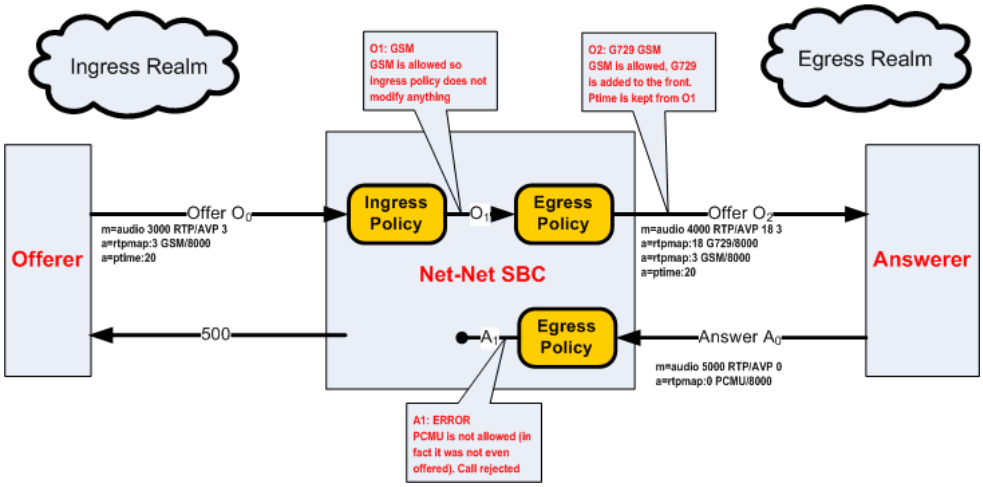


3. In the following diagram, G729 in the original SDP offer. Because once G729 is removed, no non-signaling are left in O1, thus the call is rejected.



4. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from O1 and sends this to the answerer as O2.

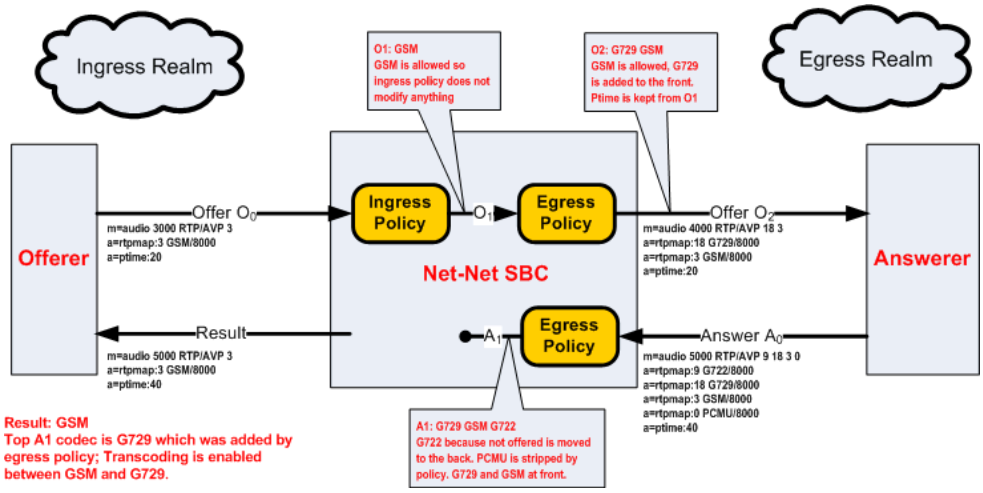
   The SDP answer states that the answerer wants to use PCMU. This is a violation of the RFC3264. Therefore the call is rejected.



   In this example, when the negotiation fails, the Oracle USM sends a 500 message to the offerer and a BYE message to the answerer.

5. In the following diagram, GSM is in the original SDP offer. It is then passed through to O1. Egress policy adds G729 and retains ptime from O1 and sends this to the answerer as O2.

   The SDP answer replies with G722 G729 GSM and PCMU. PCMU is stripped by policy, G722 is moved to the back of the answer because it was not offered. The top A1 codec was not in O1, and was added by egress policy, therefore the call is transcoded between GSM and G729.

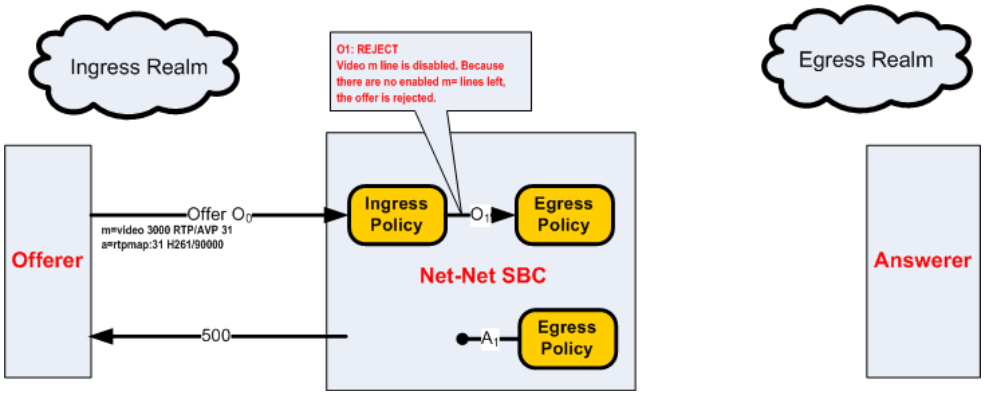


Voice Scenario 2

The following ingress and egress policies are used for scenario 2.

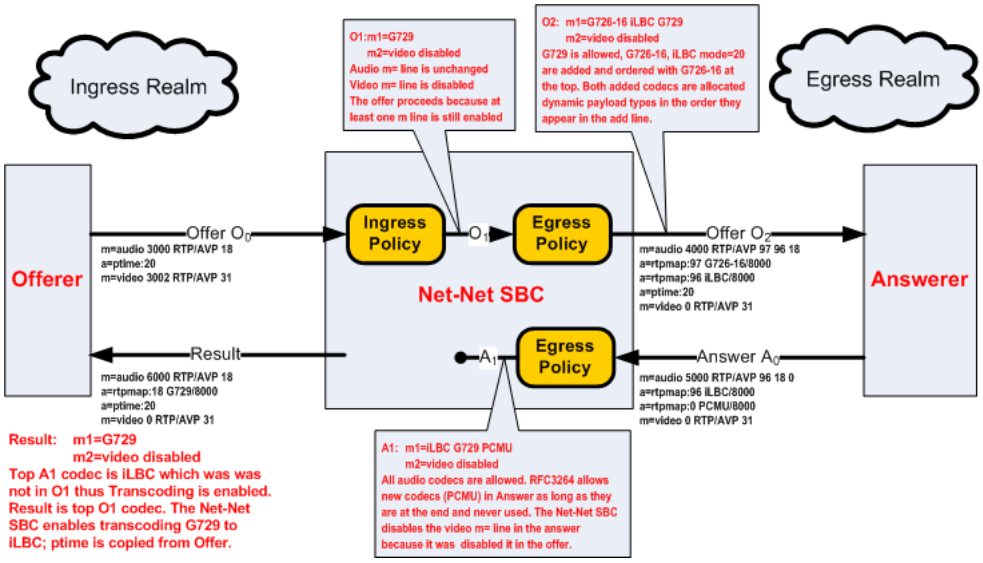
Ingress Policy		Egress Policy	
allow-codecs	Video:no PCMU:force * PCMA:force	allow-codecs	* PCMA:no
add-codecs-on-egress		add-codecs-on-egress	iLBC G726-16
order-codecs		order-codecs	G726-16 * PCMU
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

1. In the following diagram, a video m= line is offered. The ingress policy disables the video m= lines. With no enabled m= lines left, the call is rejected.



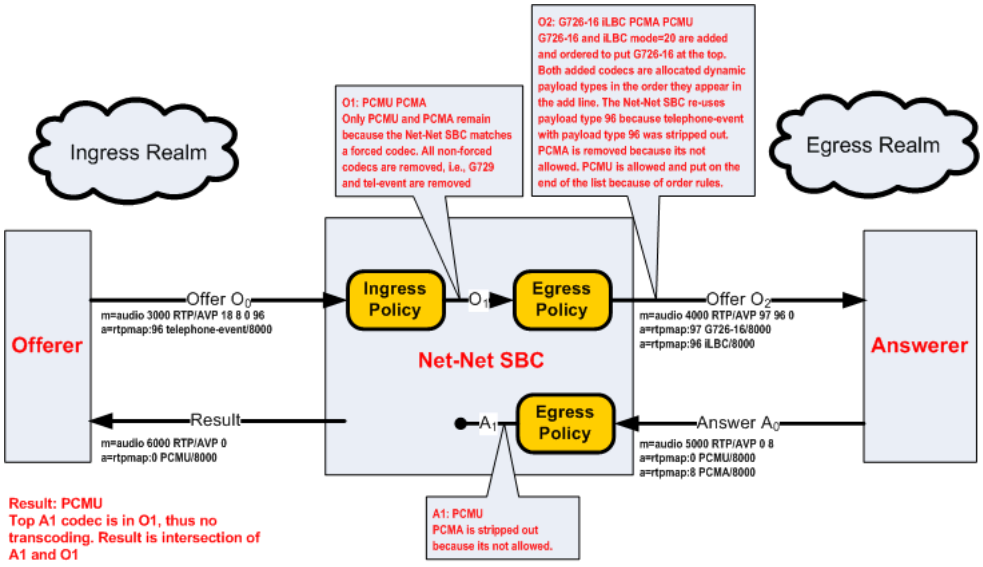
2. In the following diagram, G729 and video are offered to the Oracle USM. Ingress policy allows G729 and disables the video m= line. The egress policy adds iLBC and G726-16, and then orders the codecs according to the order-codecs parameter. The ptime is maintained between O0 and O2. Both added codecs are allocated dynamic payload types in the order they appear in their m= line. A disabled Video m= line is passed on to the answerer.

The SDP answer agreed to use iLBC, G729, and adds PCMU, and reorders them as stated. The disabled video m= line is maintained. At this point, the top codec in A1, iLBC is used and transcoded with the top codec in O1, G729.



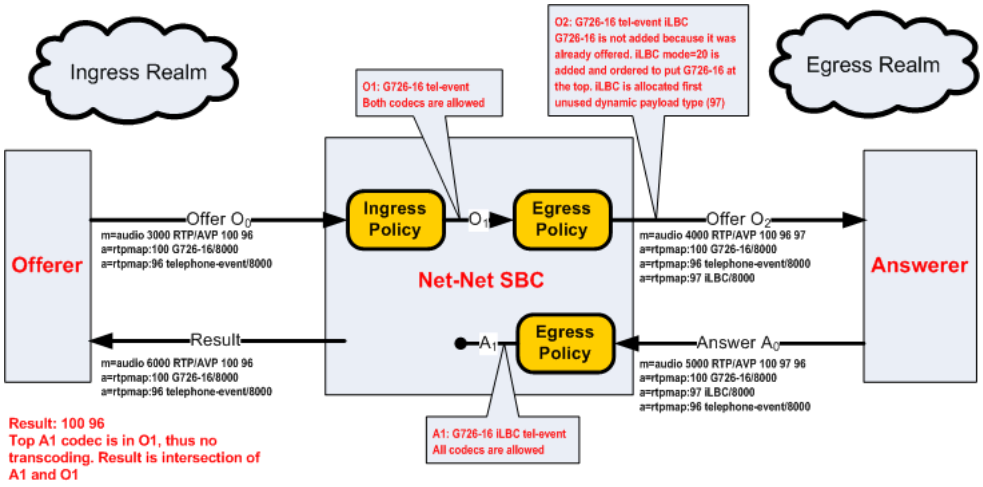
3. In the following diagram, G729 and video are offered to the Oracle USM. Ingress policy allows G729 and disables the video m= line. The egress policy adds iLBC and G726-16, and then orders the codecs according to the order-codecs parameter. The ptime is maintained between O0 and O2. Both added codecs are allocated dynamic payload types in the order they appear in their m= line.

   The SDP answer only wants to use PCMU and PCMA. The egress policy removes PCMA and passes only PCMU to the offerer. Because PCMU was in O1 and is now the only codec in A1, it is used, and no transcoding is used between the endpoints.



4. In the following diagram, G726-16 and telephone-event are offered to the Oracle USM. Ingress policy allows both codecs. The egress policy adds iLBC, and then orders the codecs according to the order-codecs parameter.

The SDP answer agreed to use all codecs, but reorders them with G726-16 in the top position. Because G726-16 is the top codec in A1, and it is also present in O1, it is used for this call without any transcoding.

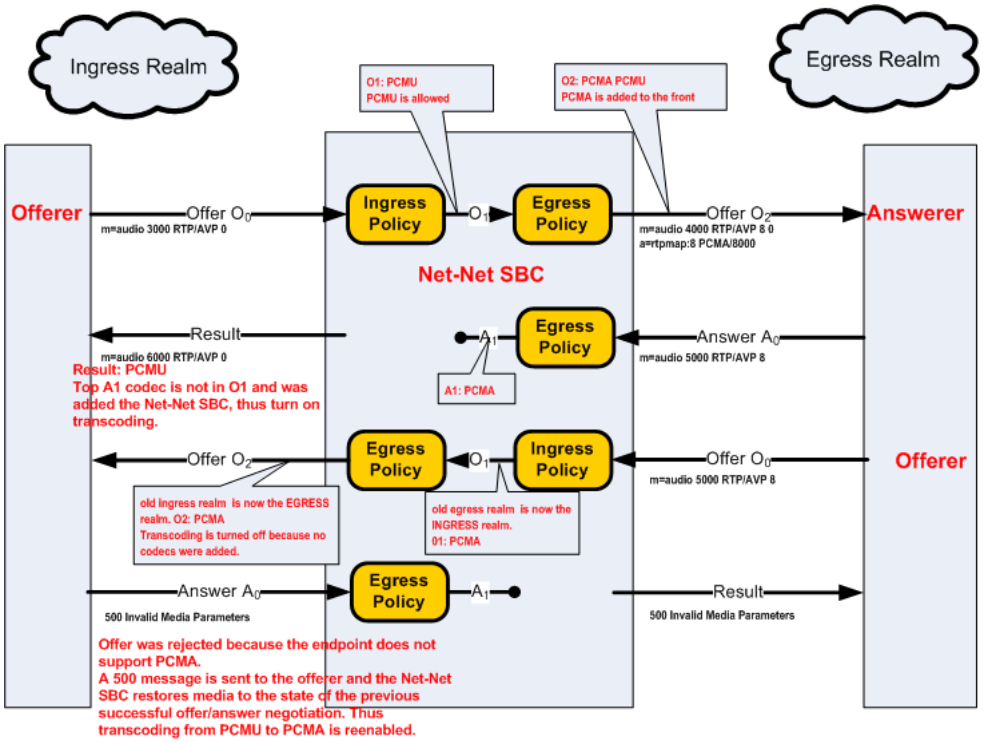


Voice Scenario 3

Voice scenario 3 involves reINVITEs. The following ingress and egress policies are used for scenario 3.

Ingress Policy		Egress Policy	
allow-codecs	PCMU G729	allow-codecs	*
add-codecs-on-egress		add-codecs-on-egress	PCMA
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

1. In the following diagram, the answerer sends a reINVITE after a previous transcoding session was established. The original offerer and answerer swap roles. The new offerer rejects the SDP offer and the call reverts to the state negotiated in the original SDP negotiation.



RFC 2833 Transcoding

RFC 2833 defines an RTP payload that functions interchangeably with DTMF Digits, Telephony Tones and Telephony Signals. The Oracle USM can monitor audio stream for in-band DTMF tones and then can convert them to data-based telephone-events, as sent in RFC2833 packets. This section explains how the Oracle USM transcodes between these RTP-based telephone events and in-band DTMF tones carried by G711. DTMF tones can only be transported in non compressed codecs. The Oracle USM supports two DTMFable non-compressed codecs: PCMU (G711µ) and PCMA (G711A).

> **Note:**
>
> The following line is added to SDP whenever telephone-event is added on egress:
> a=fmtp:101 0-15

The following two scenarios describe when telephone-event to DTMF transcoding takes place:

RFC 2833 Scenario 1

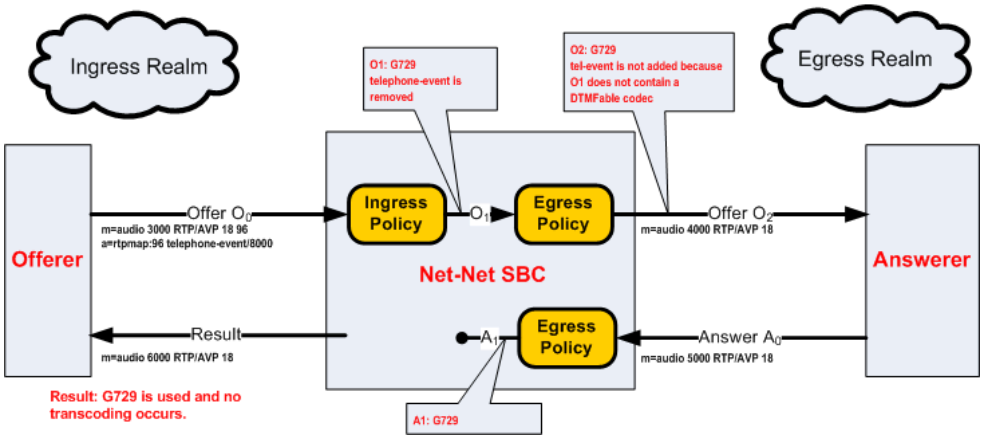
The following ingress and egress policies are used for scenario 1.

Ingress Policy		Egress Policy	
allow-codecs	* telephone-event:no	allow-codecs	* PCMA:no
add-codecs-on-egress		add-codecs-on-egress	telephone-event
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled

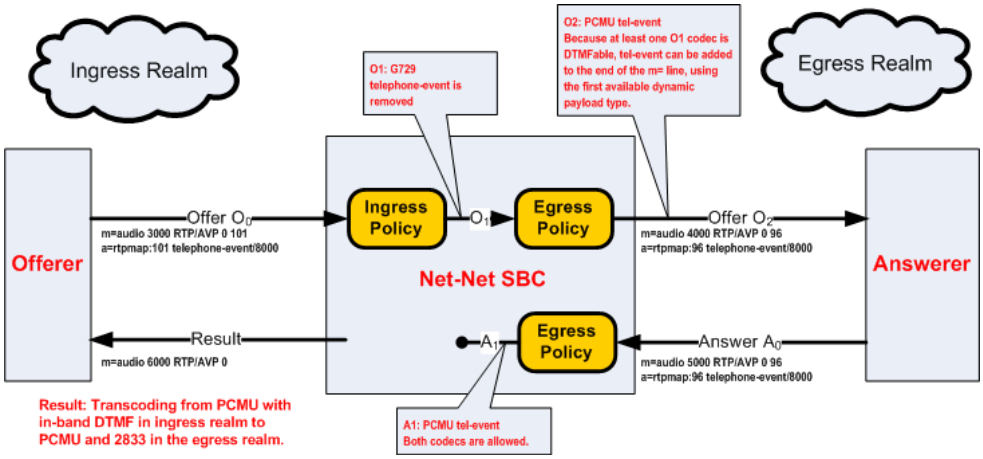
Ingress Policy		Egress Policy	
packetization-time		packetization-time	
dtmf-in-audio	preferred	dtmf-in-audio	preferred

1. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event was not added by the egress policy because the remaining audio codec in O1 was not DTMFable. G729 was the only codec forwarded on to the answerer.

   The SDP answer agreed to use the remaining audio codec, G729. A0 is unaltered by egress policy, and forwarded as the Result to the offerer. Therefore, G729 is used in both the ingress and egress realms, the call does not support RFC 2833, and the call is not transcoded.

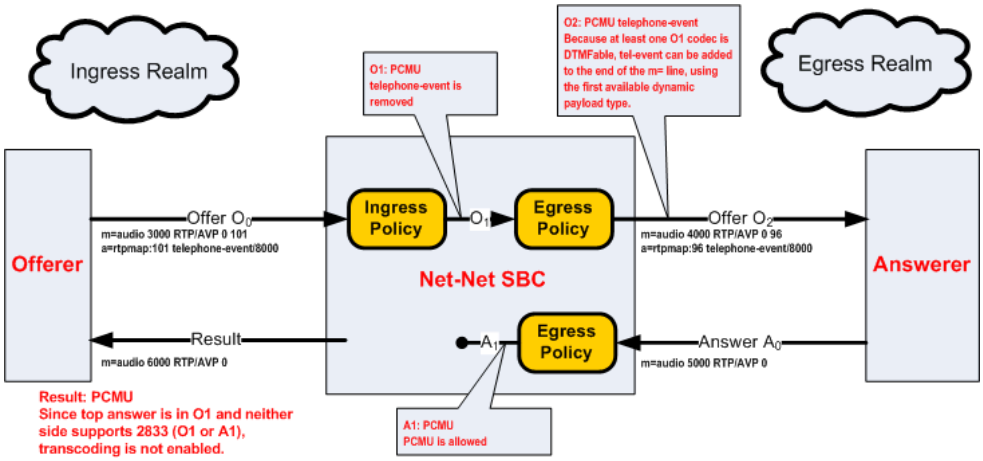


2. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event was added by the egress policy because the remaining audio codec in O1 was DTMFable. PCMU and telephone-event are then forwarded on to the answerer. Note that the telephone-event payload type is added with the lowest available dynamic type number.



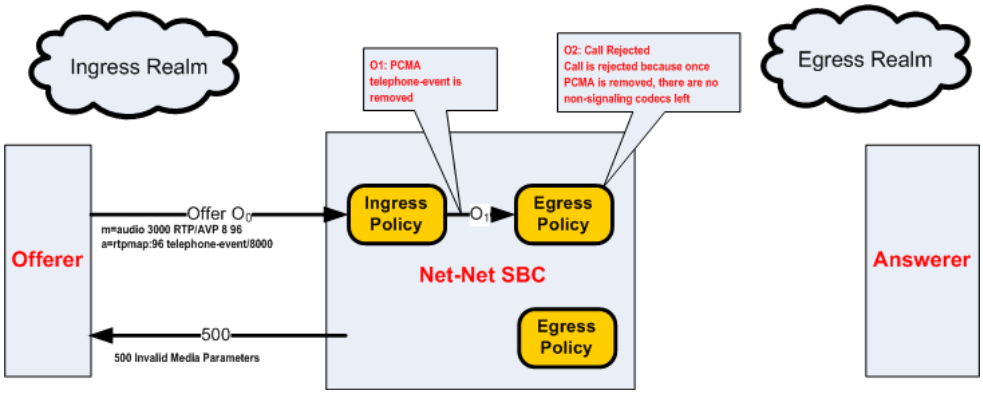
This case illustrates when the answerer supports audio and RFC 2833, but the offerer supports audio with inband DTMF. The Oracle USM transcodes between RFC2833 in the egress realm to in-band DTMF on the ingress realm.

3. In the following diagram, telephone event was offered by the offerer but was stripped by ingress policy. telephone-event is added by the egress policy because the remaining audio codec in O1 was DTMFable. PCMU and telephone-event are then forwarded on to the answerer. Note that the telephone-event payload type is added with the lowest available dynamic type number.



The SDP answer only agreed to use PCMU. When A0 reaches the egress policy, it is passed along through the Oracle USM to the offerer. Because telephone-event was not answered by the answerer and not supported in O1, it can't be used. Transcoding is therefore not used for this call.

4. In the following diagram, telephone event was offered by the offerer and was stripped by ingress policy. Since PCMA was also stripped by the egress policy, leaving no non-signaling codecs, the call is rejected. A 500 message is sent back to the offerer.



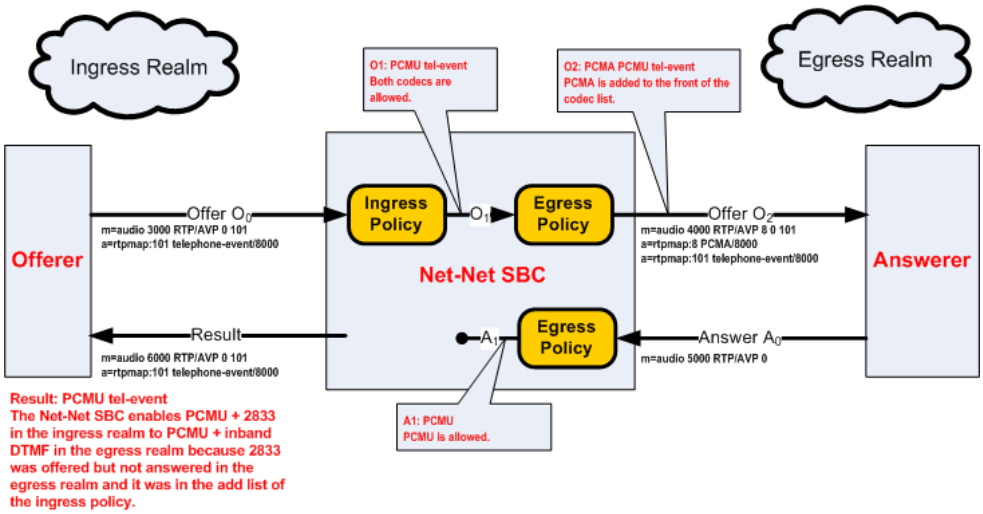
RFC 2833 Scenario 2

The following ingress and egress policies are used for RFC2833 scenario 2.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress	telephone-event	add-codecs-on-egress	PCMU
order-codecs		order-codecs	

Ingress Policy		Egress Policy	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	
dtmf-in-audio	preferred	dtmf-in-audio	preferred

1. In the following diagram, telephone event and PCMU are offered by the offerer. They are both passed to O1, and PCMA is added as it is sent to the answerer. The SDP answer, A0 disables all codecs but PCMU.

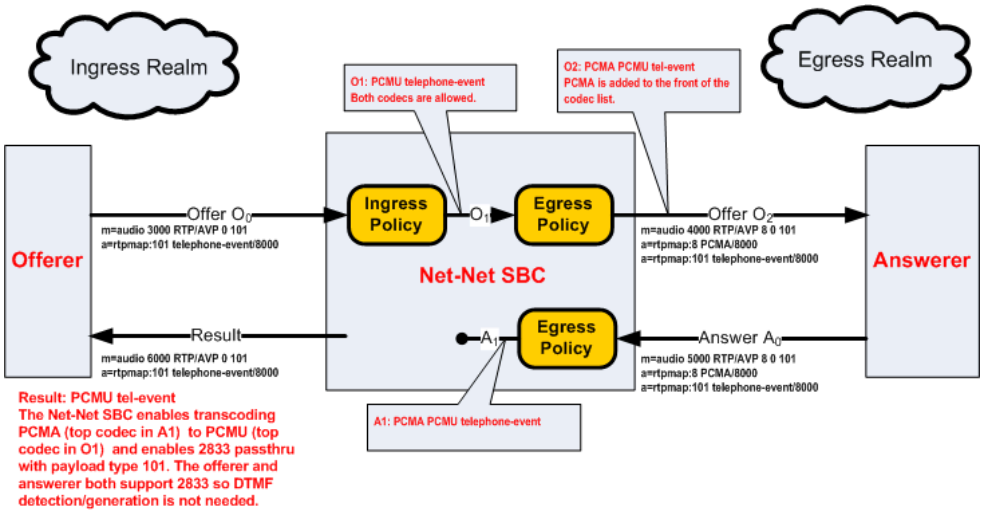


The Oracle USM adds telephone-event to the result because it is listed on the ingress policy's add-codecs-on-egress parameter and present in the offerer's SDP.

> **Note:**
>
> This is the only time the add list of an ingress policy is utilized as a check.

The result SDP includes PCMU and telephone-event in the ingress realm, which is transcoded to PCMU with in-band DTMF in the egress realm.

2. In the following diagram, telephone event and PCMU are offered by the offerer. They are both passed to O1, and PCMA is added as it is sent to the answerer. The SDP answer supports all three codecs offered, with PCMA added on top.



The answerer responds with PCMA as the preferred codec in A0. The Oracle USM compares A1 to O1 to make the transcoding decision. PCMA is the top codec in A1 and is transcoded to PCMU, the top codec in O1. Also, because telephone-event is supported by both sides of the call, it is passed through without any transcoding necessary.

This case illustrates when both endpoints are capable of sending and receiving telephone-event. Regardless of whether the audio portion of the call is transcoded, the telephone-event messages are passed through the system untouched, thus not requiring transcoding resources. This is known as telephone-event pass-through.

FAX Transcoding

FAXes are transmitted in a call as either T.30 and T.38 media. T.30 FAX is binary in-band media carried over G.711. The Oracle USM can transcode between T.38 and a faxable codec. The supported faxable codecs are PCMU and PCMA.

T.30 can only be transported in non-compressed codecs. The two non-compressed codecs supported by the Oracle USM are PCMU (G711µ) and PCMA (G711A). If a transcoding realm does not support an uncompressed codec, T.30 can not be supported in that realm. Alternatively, G711FB may be allowed specifically for FAX only.

The Oracle USM uses an internal codec called G711FB (G711 - Fall Back) that is an umbrella codec of all FAXable codecs. G711FB will default to PCMU for the purpose of offering a faxable codec. You can remap G711FB to PCMA by configuring the media-profile for it appropriately. G711FB's only use is for FAX transcoding.

FAX transcoding is triggered when you configure the add on egress parameter with either T.38 or G711FB. In a FAX scenario, when the codec policy adds either T.38 or G711FB, a new m= line is added to the SDP. When adding T.38, the new m= line specifies the T.38 codec. When adding G711FB, the new m= line specifies PCMU (or alternatively PCMA).

Once added, m= lines can not be deleted in the context of a call. The Oracle USM maintain all m= lines between itself and an endpoint throughout the course of call. All m= lines not in use can be disabled by setting their receive port to 0, but they can not be removed from the SDP.

Defining G711FB

G711 Fall Back (G711FB) is an internal codec that encompasses PCMU and PCMA for carrying fax information FAXable codecs. The G711FB codec must be configured either way

for when the Oracle USM inserts a FAXable codec in SDP. G711FB is only used for FAX transcoding scenarios.

To define G711 FB, create a media profile configuration element named g711fb and set the payload-type to 0 or 8.

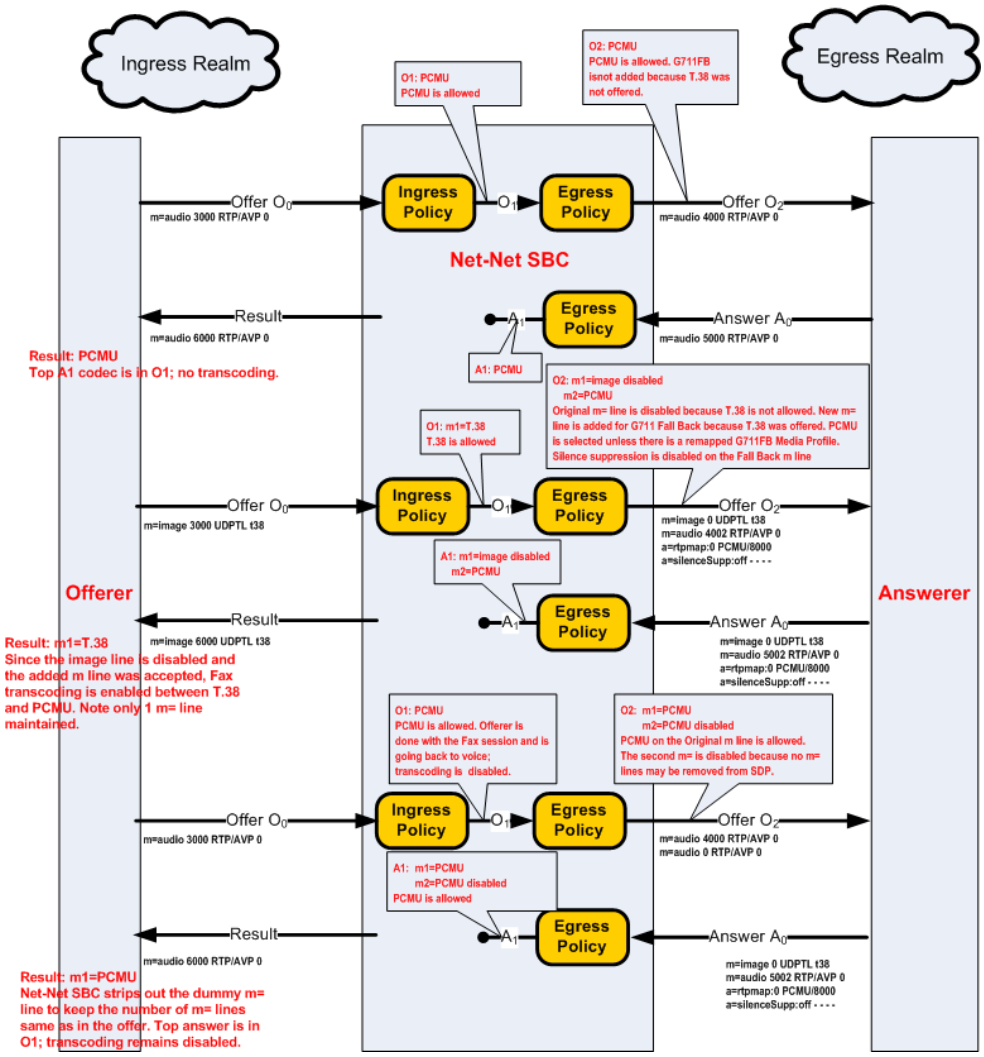
Codec (supported bit rates)	RTP Payload Type	Default Ptime (ms)	Supported Ptime (ms)
T.38	N/A	30	10, 20, 30
G711FB (64 kbps)	0, 8	30	10, 20, 30

FAX Scenario 1

The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	T.38:no
add-codecs-on-egress		add-codecs-on-egress	G711FB
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

1. In the following diagram, there are three offer-answer exchanges. Initially a PCMU-to-PCMU session is negotiated. Next, a T.38 to PCMU session is negotiated. Finally, the session reverts to non-transcoded PCMU to PCMU state.



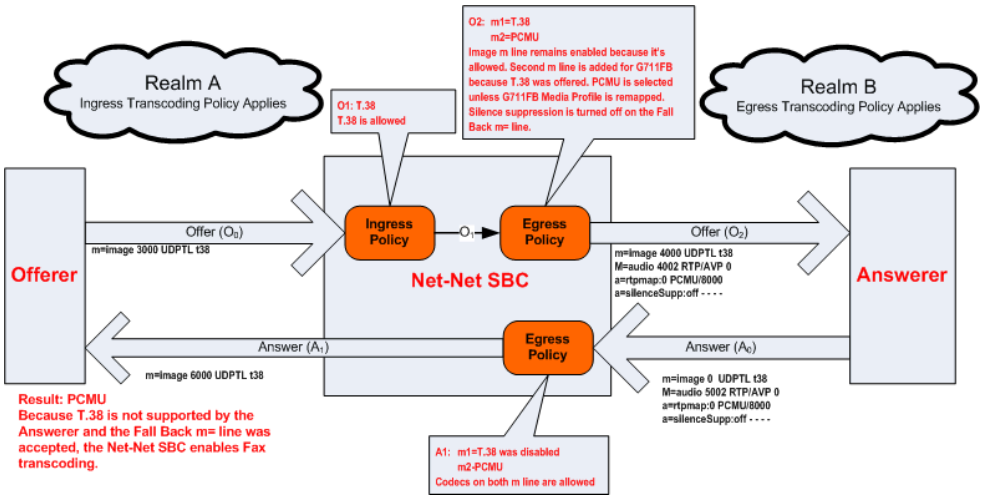
FAX Scenario 2

The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress		add-codecs-on-egress	G711FB
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

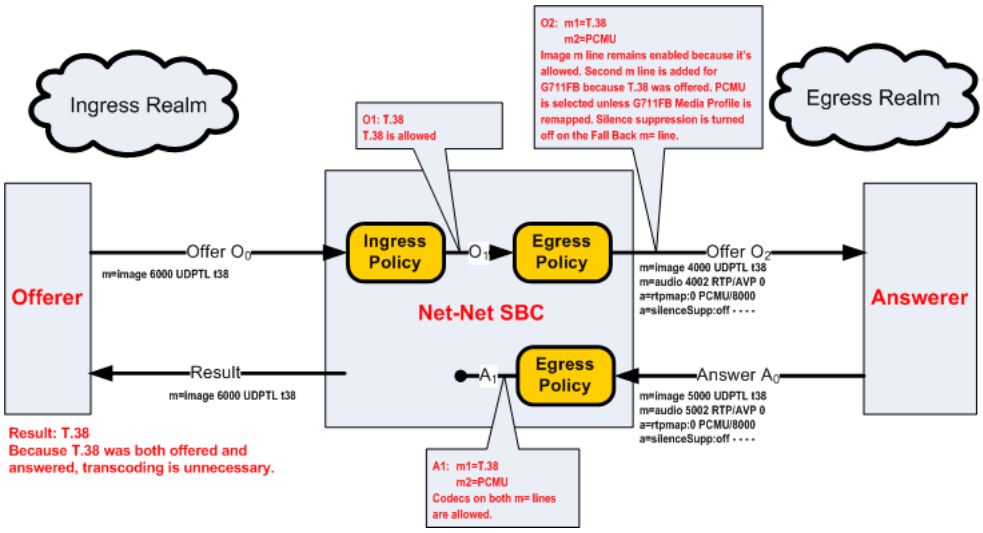
1. In the following diagram, T.38 is offered to the Oracle USM. A second m= line was added to O1 that included a G711FB codec (PCMU).

   The SDP answer agreed to PCMU, but disabled T.38. When the Oracle USM forwarded the SDP in A1 to the answerer, it stripped the second m= line. Because A1 rejects T.38 m= line, but accepts the PCMU m= line, FAX transcoding is enabled.



2. In the following diagram, T.38 is offered to the Oracle USM. A second m= line was added to O1 that included a G711FB codec (PCMU).

   The SDP answer agreed to PCMU and T.38. Because both O1 and A1 support T.38, the call proceeds without transcoding.



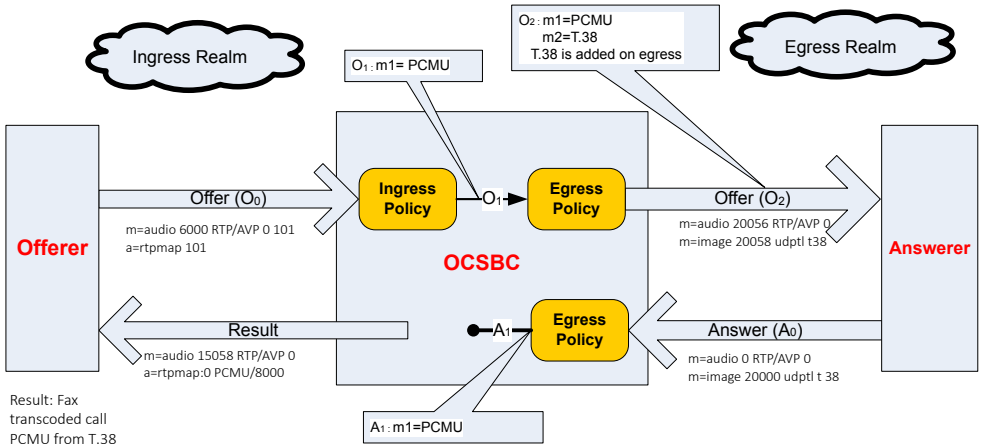
FAX Scenario 3

The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress		add-codecs-on-egress	PCMU, G729 ,T.38
order-codecs		order-codecs	
force-ptime	disabled	force-ptime	disabled
packetization-time		packetization-time	

1. In the following diagram, PCMU and telephone-event codecs are received by Oracle USM .The egress codec policy has PCMU, G729 and T.38 **add-codecs-on-egress**.

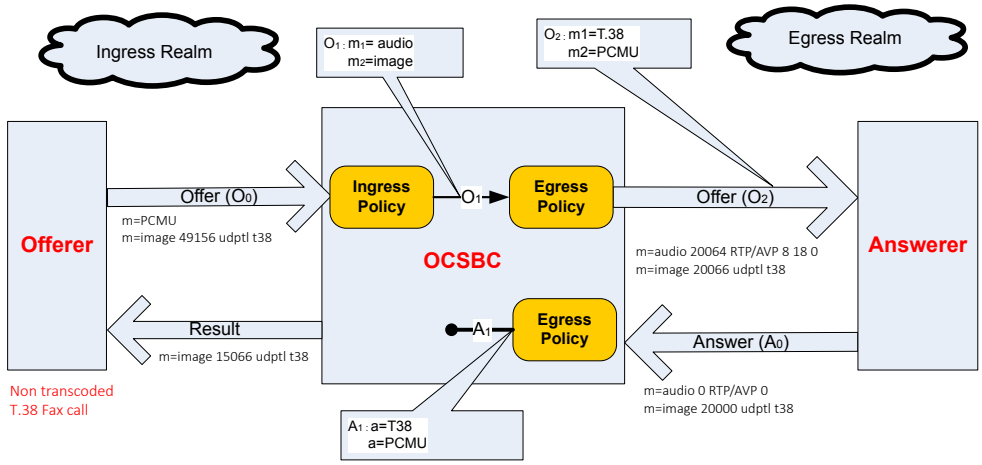
Since there is a faxable codec in the SDP offer and T.38 in **add-on-egress**, the non-faxable codec G729 is stripped and T.38 is added to egress offer.
In the A0 answer the audio m-line is zero indicating disabled and the port for the image m-line is non-zero (20000) so the called party has selected T.38. Hence PCMU in realm A is transcoded to T.38 in realm B.



2. In the following diagram, PCMU and T38 are received by the Oracle USM. The egress codec policy has PCMU, G729, and T.38 **add-codecs-on-egress**.

   T.38 is present in the O0 offer, and therefore will not be explicitly added by the egress codec policy to the O2 Offer. The logic to remove non-faxable codecs is only invoked when T.38 is added by the **add-codecs-on-egress** parameter. In this example, T.38 is not added since is already present in SDP. With PCMU and T.38 received, G729 as a non-faxable codec is not removed. Therefore, PCMU, T.38, and G729 are all present in the O2 SDP offer sent to the answerer.

   In the A0 answer, the audio m-line port is 0 indicating that audio is disabled, and the port for the image m-line is nonzero (20000), thus the called party has selected T.38. In the A1 answer the OCSBC send the audio m-line port as 0 indicating that audio is disabled, and the port for the image m-line is nonzero (20000). This set-up results in a non-transcoded T. 38 -OCSBC -T.38 fax call.



Transrating

The Oracle USM can transrate media as it exits the Oracle USM into the network. Transrating is also known as forced packetization time (ptime), and is used to enforce a configured ptime within a realm. Transrating is often desirable when devices in a realm can only accept media with a specific ptime, or to optimize bandwidth.

If this feature is configured, the media portion of a call is transrated regardless of which codecs are ultimately chosen for each realm as long as they are transcodable. This allows realms that have devices that can only use a single packetization interval to interwork with devices that may or may not have the same packetization capabilities.

You must enable force-ptime in the egress codec policy and then specify the packetization time to force. When force ptime is enabled, it implicitly masks all codecs not of the specified packetization time that are listed in that codec policy's allow codecs and add codecs on egress parameters. For example, if force ptime is enabled with a packetization time of 20 ms, then no G723 codecs (which are only available at 30, 60, and 90 ms) may be active via codec policy in that realm.

Transrating occurs when forced-ptime is enabled and the offered and answered ptimes do not match and the top non-Signaling codec of A1 and top non Signaling codec of O1 are Transcodable.

> **Note:**
>
> Answered ptime A1 does not have to be equal to the ptime inserted into the outgoing offer O2, it just has to be different than the offer the Oracle USM received (O1).

Please refer to Transcodable Codecs for current list of ptimes per codec supported by the DSPs.

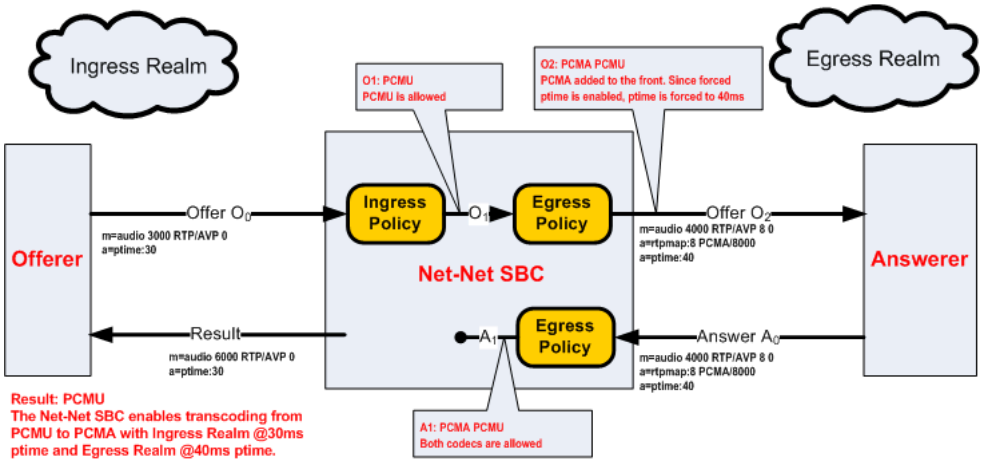
Transrating Scenario 1

The following ingress and egress policies are used for this FAX scenario.

Ingress Policy		Egress Policy	
allow-codecs	*	allow-codecs	*
add-codecs-on-egress		add-codecs-on-egress	PCMA
order-codecs	G723 *	order-codecs	
force-ptime	disabled	force-ptime	enabled
packetization-time		packetization-time	40

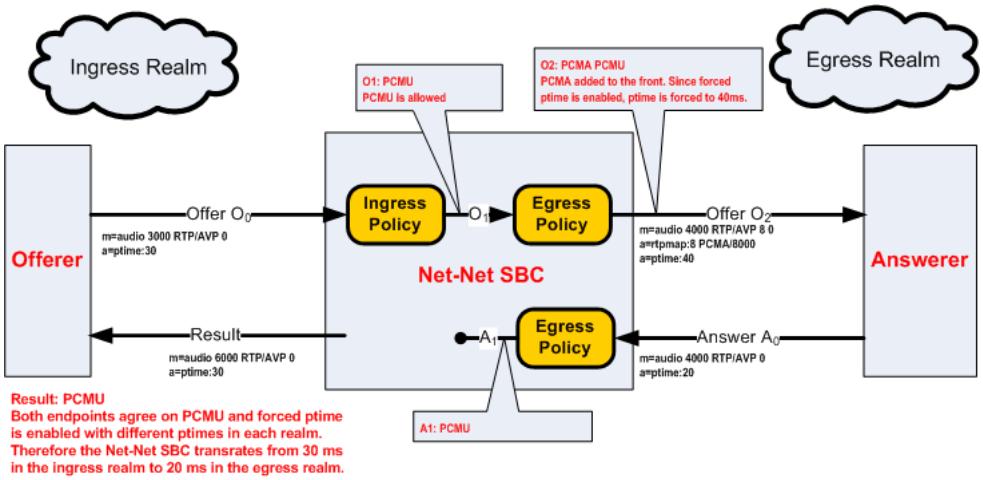
1. In the following diagram, PCMU is offered in the ingress realm with 30ms ptime, and the egress realm is forced to use 40ms ptime. PCMA is added as the top codec for the egress realm.

   The Oracle USM enables transcoding between the ingress realm (PCMU) and the egress realm (PCMA) and the ptimes as negotiated are also maintained.



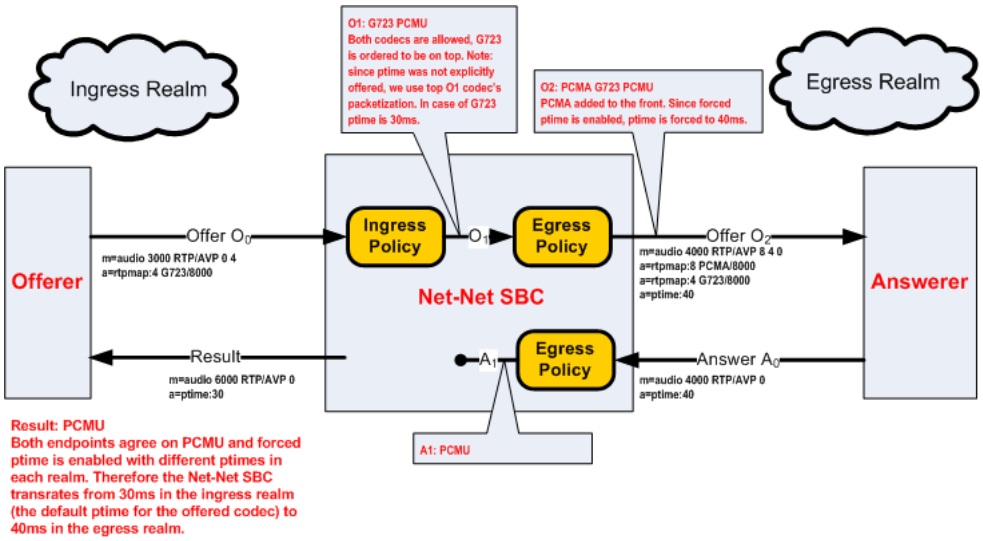
2. In the following diagram, PCMU is offered in the ingress realm with a ptime of 30ms, and forced to 40 ms in the egress realm by policy.

   The answerer chooses to use PCMU with a 20 ms ptime. Thus the call is not transcoded, but it is transrated from 30ms in the ingress realm to 20ms in the egress realm.



3. In the following diagram, PCMU and G723 are offered in Realm A. The top codec's ptime (30ms) is implied as the one for the ingress realm. The Oracle USM adds PCMA to the SDP offer with a 40ms ptime.

   The answerer chooses to use PCMU with a 40 ms ptime. Thus the call is transrated from 30ms in the ingress realm to 40ms in the egress realm.



Default Media Profiles

The Oracle USM contains a set of default media profiles that define characteristics of well-known IANA codecs. You can not view the default media profiles' configurations, but you can override them by configuring identically-named media profile configuration elements.

Transcodable codecs are a subset of the default media profiles which the Oracle USM can transcode between.

Transcodable Codecs

The following list shows the transcodable codecs which the Oracle USM can add to SDP. These codecs all reflect default media profiles for their given names. Enter codecs in the configuration exactly as below.

- PCMU
- PCMA
- G729
- G729A
- iLBC
- telephone-event
- T.38
- G711FB
- G726
- G726-16
- G726-24
- G726-32
- G726-40
- G722

- G723
- GSM
- AMR
- AMR-WB
- EVRC0
- EVRC
- EVRC1
- EVRCB0
- EVRCB
- EVRCB1

When creating an override media profile from the previously listed codec, case is ignored. Also, GSM is GSM-FR.

Preferred Default Payload Type

When the Oracle USM adds a codec with a dynamic payload type to SDP, it uses the lowest unused payload number. You can configure a preferred payload type for a dynamic codec by creating an override media profile. This makes the Oracle USM use your preferred payload type for insertion into SDP. If you configure a dynamic codec to use a preferred payload type, and that payload type is already in use, the codec will still be inserted into SDP, but with the first available dynamic payload type.

For example, you create a media profile for telephone-event with a payload type of 101. If telephone-event is added to SDP, and payload type 101 is already in use in the SDP, the Oracle USM will use the first available payload type in the 96-127 range when adding telephone-event.

Redefining Codec Packetization Time

You can configure a media profile with a packetization time (ptime) that overrides the codec's default ptime. Transcoding functions look up and use default ptimes when not specified in offered or answered SDP. Default ptime for most audio codecs is 20ms; some however are 30ms. See the Transcodable Codecs list for default values.

To change the default ptime for a codec, you must create a media profile that overwrites the default ptime parameter with your new packetization time. When SDP is received with no 'a= ptime' attribute or when adding the codec to egress SDP, the newly configured ptime is used.

New default ptime for a media profile is entered by typing "ptime=<x>" in the parameters parameter, where <x> is the new default packetization time.

mptime Support for Packet Cable

The SDP specification lacks the ability to specify unique packetization times per codec when more than one codec is listed in an m= line. The ptime attribute is not related to a specific codec but to the entire m= line. When multiple codecs appear on a single m= line, the PacketCable mptime attribute can specify different packetization times for each codec.

The Oracle USM adheres to PKT-SP-NCS1.5-I01-050128 and PKT-SP-EC-MGCP-I06-021127 for processing and generating mptime. The mptime line uses an integer to indicate the

packetization time for each corresponding codec in the m= line. The dash character, "-", on an mptime line is used for non-packetized codecs, such as CN or telephone-event.

If the Oracle USM receives an invalid mptime, it is ignored and removed. If a valid mptime is received in the incoming SDP, its values will be used for packetization times of each corresponding codec and a valid mptime line will be sent in the outgoing SDP.

Valid:

```
m=audio 10000 RTP/AVP 0 96 8
a=mptime:20 - 30
a=rtpmap:96 telephone-event/8000
```

Valid: 'ptime' attribute is ignored

```
m=audio 10000 RTP/AVP 0 8
a=mptime:20 30
a=ptime:30
```

Invalid: dash cannot be first mptime value

```
m=audio 10000 RTP/AVP 96 0
a=mptime: - 20
```

When Oracle USM includes an mptime in an outgoing SDP, it will also always add a ptime attribute with the value of the most preferred codec. This is done to increase the interoperability with devices that do not support mptime.

AMR-NB and AMR-WB Specifications

The Oracle USM supports Adaptive Multi-Rate Narrow Band & Wide Band codecs. All configurations of this codec, as indicated by SDP, are transcodable except when the following SDP parameters are enabled:

- robust-sorting
- interleaving

When AMR is configured in a codec policy's add-codecs-on-egress parameter, it is forwarded from the Oracle USM with the following default settings:

- 12.2 kbps (AMR-NB)
- 23.85 kbps (AMR-WB)
- RTP/IF1 format
- No redundant packets
- bandwidth efficient default payload
- No CRC frame
- 20ms default ptime

> **Note:**
>
> AMR and AMR-WB each require a separate license.

EVRC Family of Codecs

The Acme Packet 6300 supports 2 codecs in the EVRC family. Each codec has has three variants.

- EVRC-A is also commonly referred to as EVRC. The following EVRC-A packet formats are supported:

Header-free packet format = EVRC0

Bundled/interleaved packet format = EVRC

Compact Bundled packet format = EVRC1

- The following EVRC-B packet formats are supported:

Header-free packet format = EVRCB0

Bundled/interleaved packet format = EVRCB

Compact Bundled packet format = EVRCB1

EVRC-A Codec for Transcoding

The Acme Packet 6300 supports the Enhanced Variable Rate codec (EVRC) for transcoding. Three subtypes of the EVRC codec are supported as media-profiles:

- EVRC0—header-free EVRC format. This codec is transcodable.
- EVRC—non-header-free EVRC format. This codec is transcodable.
- EVRC1—fixed rate EVRC format. This codec is transcodable.

To transcode to and from EVRC, the EVRC license must be installed.

> **Note:**
>
> The Acme Packet 6300 handles optional parameters according to RFC 4788 unless otherwise specified.

EVRC0 Supported Options

Required Parameters: none

Optional Parameters:

- silencesupp—If this parameter is not present in a DTX session, the default value 1 MUST be assumed.
- dtxmax—If this parameter is not present in a DTX session, the default value is 32.
- dtxmin—If this parameter is not present in a DTX session, the default value is 12.
- hangover—If this parameter is not present in a DTX session, the default value 1 MUST be assumed.

The payload type is dynamic for this codec. EVRC0 specifies the header-free format of the EVRC codec. Only a single frame (20 ms) is allowed in header-free mode.

EVRC Supported Options

Required Parameters: none

Optional parameters:

- ptime
- maxptime
- maxinterleave—If not signaled, the maxinterleave length is set to 5.
- silencesupp—If this parameter is not present, the default value 1 MUST be assumed.
- dtxmax—See dtxmax in EVRC0 section.
- dtxmin—See dtxmin in EVRC0 section.
- hangover—See hangover in EVRC0 section.

The payload type is dynamic for this codec. The default ptime is 20 ms. The ptimes 20, 40, and 60 ms are supported for transcoding.

EVRC1 Supported Options

Required parameters: none

Optional parameters:

- ptime—See RFC 4566
- maxptime
- fixedrate—Valid values are 0.5 or 1. If this parameter is not present, 0.5 is assumed.
- silencesupp—If this parameter is not present, 1 MUST be assumed.
- dtxmax—See dtxmax in EVRC0 section.
- dtxmin—See dtxmin in EVRC0 section.
- hangover—See hangover in EVRC0 section.

The payload type is dynamic for this codec. Only the 20 ms ptime is supported for transcoding. The media rate will be fixed to either full or half rate depending on the fixedrate parameter (half rate is default).

Default settings for EVRC encoding

- CDMA Rate change for Dim and Burst disabled
- CDMA DTX control enabled

EVRC Configuration

In the ACLI, refer to EVRC codecs exactly as follows:

```
EVRC0
EVRC
EVRC1
```

EVRC-B Codec for Transcoding

The Acme Packet 6300 supports the Enhanced Variable Rate codec, extension B (EVRCB) for transcoding. Three subtypes of the EVRCB codec are supported as media-profiles:

- EVRCB0—This codec is transcodable.
- EVRCB—This codec is transcodable.
- EVRCB1—This codec is transcodable.

To transcode to and from EVRCB, the EVRCB license must be installed.

> **Note:**
>
> The Acme Packet 6300 handles optional parameters according to RFC 4788 unless otherwise specified.

EVRCB0 Supported Options

EVRBC0 is the header-free format of the EVRCB codec. Only a single frame (20 ms) is allowed in header-free mode.

Required Parameters: none

Optional Parameters:

- silencesupp — If this parameter is not present in a DTX session, the default value 1 MUST be assumed.
- dtxmax — If this parameter is not present in a DTX session, the default value is 32.
- dtxmin — If this parameter is not present in a DTX session, the default value is 12.
- hangover — If this parameter is not present in a DTX session, the default value 1 MUST be assumed.

The payload type is dynamic for this codec. EVRCB0 specifies the header-free format of the EVRCB codec. Only a single frame (20 ms) is allowed in header-free mode.

EVRCB Supported Options

Required Parameters: none

Optional parameters:

- ptime
- maxptime
- maxinterleave—If not signaled, the maxinterleave length is set to 5.
- silencesupp—If this parameter is not present, the default value 1 MUST be assumed.
- dtxmax
- dtxmin
- hangover

The payload type is dynamic for this codec. The default ptime is 20 ms. Only the 20 ms ptime is supported for transcoding.

EVRCB1 Supported Options

Required parameters: none

Optional parameters:

- ptime—See RFC 4566
- maxptime
- fixedrate—Valid values are 0.5 or 1. If this parameter is not present, 0.5 is assumed.
- silencesupp—If this parameter is not present, 1 MUST be assumed.
- dtxmax
- dtxmin
- hangover

The payload type is dynamic for this codec. Only the 20 ms ptime is supported for transcoding. The media rate will be fixed to either full or half rate depending on the fixedrate parameter (half rate is default).

Default fixed settings for EVRCB encoding

- CDMA Rate change for Dim and Burst disabled
- CDMA DTX control enabled

EVRCB Configuration

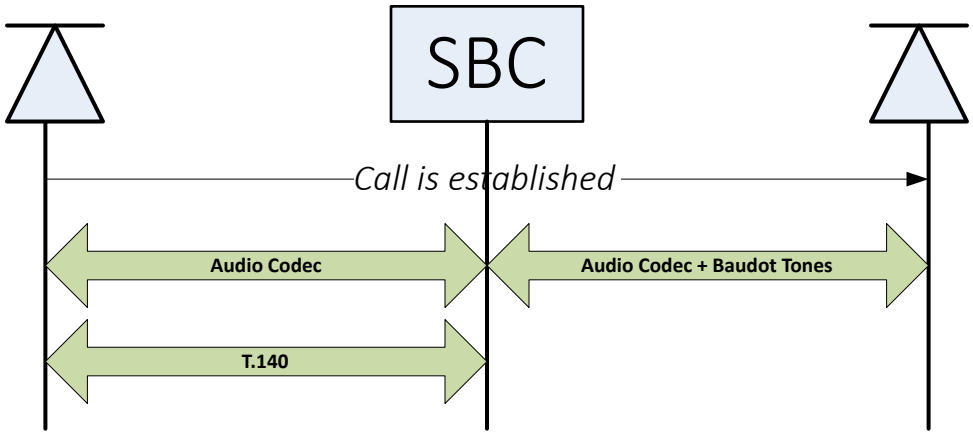
In the ACLI, refer to EVRCB codecs exactly as follows:

```
EVRCB0
EVRCB
EVRCB1
```

T.140 to Baudot Relay

The T.140 to Baudot Relay feature uses the Oracle USM's transcoding resources to relay T.140 text messages to Baudot tones and vice versa. The T.140 Protocol is used for multimedia text conversation over IP and is designed as a replacement for TDD devices. Baudot tones are a common protocol in the US in Telecommunications Device for the Deaf (TDD). Details of the protocol implementation are available in TIA/EIA-825-A. The T.140-Baudot relay is a regulatory requirement, and is specified for both emergency and non-emergency traffic. This feature is only available on Acme Packet 4600 and 6300 with transcoding hardware.

T.140 to Baudot transcoding entails that one call leg is provisioned with a Baudot-capable codec, and the other call leg accepts T.140 in the SDP. Additionally, the T.140 side includes an audio stream. Once this scenario is established, the call may begin as audio-to-audio. At any point forward, T.140 may be received on one call leg or Baudot tones may be received on the other call leg. Each text indication will be transcoded to its compliment on the other side of the call. When T.140 <-> Baudot tone transcoding is active, the existing audio stream is preempted.



The system's transcoding hardware detects baudot tones in the incoming audio stream and generates T.140 packets on an outbound text stream. In the reverse direction T.140 packets will be detected on the text stream and Baudot tones will be generated on the appropriate outgoing audio stream.

T.140 to Baudot relay is invoked when an outbound **codec-policy** removes any text "m=" line from the egressing offer. The processing also removes all non-Baudot-tone capable codecs from the egress SDP offer. If at this point no Baudot-capable codecs remain in the SDP, the call is torn down.

Baudot Tones capable codecs:

- PCMA
- PCMU
- EVRC
- EVRC0
- EVRC1
- EVRCB
- EVRCB0
- EVRCB1

Codec Policy Configuration for T.140 to Baudot Relay

To support T.140 to Baudot relay, configure the **allow-codecs** parameter in the **codec-policy** with text:no. This value causes the Oracle USM to strip any "m=text" occurrence in the outbound INVITE and enable T.140 to Baudot transcoding. When SDP passes through the Oracle USM and a text "m=" line is removed on the egress side of the call, then T.140-baudot relay/transcoding will be invoked for that call.

Unlike other transcodable codecs, T.140 is not valid in **add-codecs-on-egress** parameter.

Limitations

The following scenarios are not supported:

- Configuration of T.140 in **add-codes-on-egress**
- Hairpin calls (T.140 - Baudot Relay - T.140)

- Lawful Intercept
- SRTP for T.140

T.140 to Baudot Relay Examples

This section includes two examples; access-side call initiation and core-side call initiation. The following codec-policy is applied to the UE-side realm.

codec-policy	
name	T140-to-tones
allow	text:no *
add-codecs-on-egress	PCMU
non-relevant parameters omitted	

Figure 19-1 Call Initiated from the Access-side

The call is set up so that the Oracle USM will transcode audio between AMR and PCMU, and seamlessly relay T.140 to and from Baudot tones.

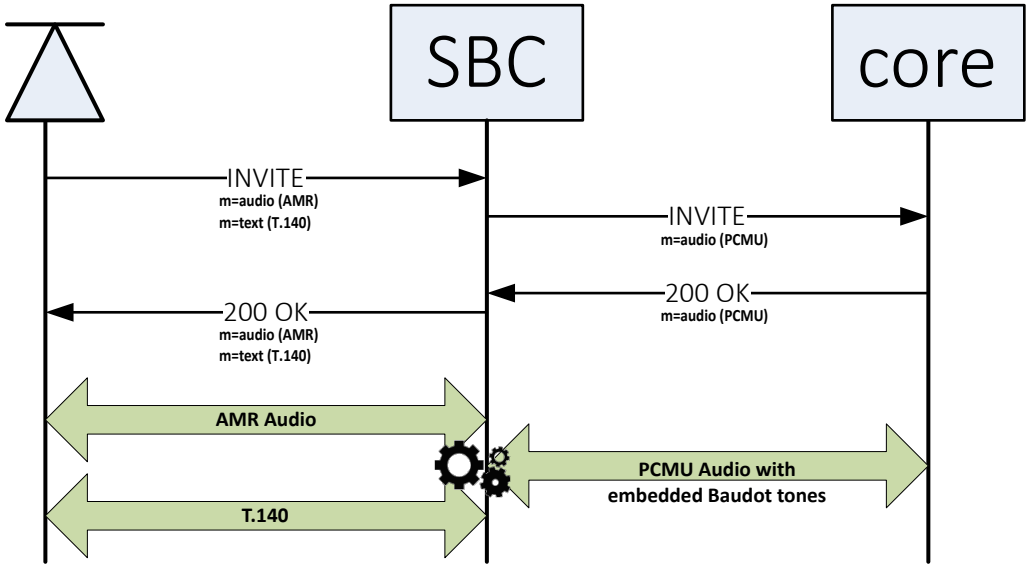
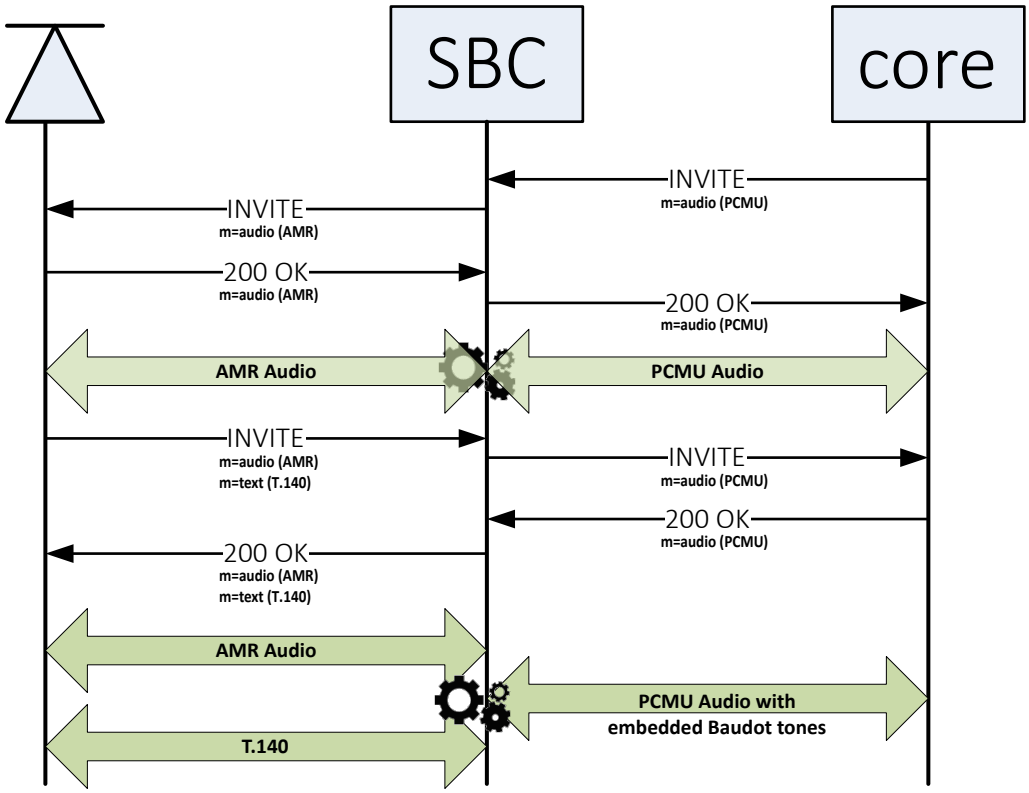


Figure 19-2 Call Initiated fom the Core with T.140 reINVITE



Opus Codec Transcoding Support

Opus is an audio codec developed by the IETF that supports constant and variable bitrate encoding from 6 kbit/s to 510 kbit/s and sampling rates from 8 kHz (with 4 kHz bandwidth) to 48 kHz (with 20 kHz bandwidth, where the entire hearing range of the human auditory system can be reproduced). It incorporates technology from both Skype's speech-oriented SILK codec and Xiph.Org's low-latency CELT codec. This feature adds the Opus codec as well as support for transrating, transcoding, and pooled transcoding to the 4600 and 6300 platforms.

Opus can be adjusted seamlessly between high and low bit rates, and transitions internally between linear predictive coding at lower bit rates and transform coding at higher bit rates (as well as a hybrid for a short overlap). Opus has a very low algorithmic delay (26.5 ms by default), which is a necessity for use as part of a low audio latency communication link, which can permit natural conversation, networked music performances, or lip sync at live events. Opus permits trading-off quality or bit rate to achieve an even smaller algorithmic delay, down to 5 ms. Its delay is very low compared to well over 100 ms for popular music formats such as MP3, Ogg Vorbis, and HE-AAC; yet Opus performs very competitively with these formats in terms of quality across bit rates.

Transcoding the Opus codec requires a special license as it is subject to a royalty agreement. Licensing supports up to the full density for this codec in bins of 25. A feature bit is required as well as a field for the capacity limit. The capacity limit is stored in 12 bits per codec allowing up to 102,375 licensed sessions. This limit is sufficient for future hardware iterations' projected session densities.

Opus Supported Options

Required SDP Parameters:

rate — Specifies the sampling frequency. This parameter is mapped to the RTP clock rate in “a=rtpmap”. The range is limited to and must be 48000 Hz.

Optional SDP Parameters:

- **maxplaybackrate** — Specifies the maximum output sampling rate in Hz that the receiver is capable of rendering. The range is 8 kHz to 48 kHz; common values are 8, 12, 16, 24, and 48 kHz.
- **sprop-maxcapturerate** — Specifies the maximum input sampling rate in Hz that the sender is likely to produce. The Vocallo OCT2224 DSP currently supports only 8000 and 16000 Hz for transcoding. The range is 8 kHz to 48 kHz; common values are 8, 12, 16, 24, and 48 kHz.
- **ptime** — Specifies the packetization interval in milliseconds. The DSP supports packetization intervals of 10, 20, 40, 60, 80, and 100 ms. This parameter is mapped to “a=ptime” in the SDP. Possible values are 3, 5, 10, 20, 40, 60, or an arbitrary multiple of Opus frame sizes rounded up to the next full integer value up to a maximum value of 120. The default is 20 ms.
- **maxptime** — Specifies the maximum packetization interval allowed. The default is 100 ms.
- **minptime** — Specifies the minimum packetization interval allowed. The default is 20 ms.
- **maxaveragebitrate** — Specifies the maximum average rate of bits received for a session in bits per second. Although the range is 6000 to 51000, only bit rates of 6000 to 30000 bps are transcodable by the DSP. A media profile configured with a value for **maxaveragebitrate** greater than 30000 is not transcodable and cannot be added on egress in the **codec-policy** element.
- **stereo** — Specifies whether the decoder receives stereo or mono signals. The possible values are 0 (mono) and 1 (stereo). The default is 0.
- **sprop-stereo** — Specifies whether the sender is likely to produce stereo audio. The possible values are 0 (mono) and 1 (stereo). The default is 0.
- **cbr** — Specifies whether the decoder uses a constant or a variable bit rate. The possible values are 0 (variable bit rate) and 1 (constant bit rate). The default is 0.
- **useinbandfec** — Specifies whether the Opus decoder supports Forward Error Correction (FEC). The possible values are 0 (no) and 1 (yes). The default is 1.
- **usedtx** — Specifies whether the Opus decoder utilizes Discontinuous Transmission (DTX). The possible values are 0 (no) and 1 (yes). The default is 0.

The payload type is dynamic for this codec.

Sample media-profile configuration for adding Opus

Parameter	Value
name	opus
subname	WB
media-type	audio
payload-type	104
transport	RTP/AVP

Parameter	Value
clock-rate	48000
req-bandwidth	0
frames-per-packet	0
parameters	maxplaybackrate=16000 sprop-maxcapturerate=16000 usedtx=0
average-rate-limit	5000
peak-rate-limit	0
max-burst-size	0
sdp-rate-limit-headroom	0
sdp-bandwidth	enabled
police-rate	0
standard-pkt-rate	0

Monitoring and Debugging

CLI commands:

The **show sipd codecs** command is modified to add **opus Count**.

SNMP:

- New SNMP OID **apSysXCodeOPUSCapacity** is added to transcoding utilization statistics as reported in the **apSysMgmtGroupTrap**. When utilization falls below 80%, the **apSysMgmtGroupClearTrap** is sent.
- Opus realm statistic **apCodecRealmCountOPUS** is added to **apCodecRealmStatsEntry**.

Alarms:

Licensed Opus Transcoding Capacity Threshold Alarm — A warning level alarm that doesn't affect health is triggered when the Opus transcoding utilization exceeds 95% of licensed capacity. The alarm is cleared when the Opus transcoding utilization falls below 80% of licensed capacity.

SILK Codec Transcoding Support

SILK is an audio codec developed by Skype Limited that supports bit rates from 6 kbit/s to 40 kbit/s and sampling rates of 8, 12, 16, or 24 kHz. It can also use a low algorithmic delay of 25 ms (20 ms frame size + 5 ms look-ahead). This feature adds the SILK codec as well as support for transrating, transcoding, and pooled transcoding to the 4600 and 6300 platforms.

Transcoding the SILK codec requires a special license as it is subject to a royalty agreement. Licensing supports up to the full density for this codec in bins of 25. A feature bit is required as well as a field for the capacity limit. The capacity limit is stored in 12 bits per codec allowing up to 102,375 licensed sessions. This limit is sufficient for future hardware iterations' projected session densities.

SILK Supported Options

Required SDP Parameters:

rate — Specifies the sampling frequency. SILK supports four different audio bandwidths – narrowband at 8 kHz, mediumband at 12 kHz, wideband at 16 kHz, and super wideband at 24 kHz. This parameter is mapped to the RTP clock rate in "a=rtpmap". The DSP only supports audio sampling rates of 8 kHz and 16 kHz for transcoding; the 12 kHz and 24 kHz bandwidths are not transcodable.

Optional SDP Parameters:

- **ptime** — Specifies the packetization interval in milliseconds. The DSP supports packetization intervals of 20, 40, 60, 80, and 100 ms. This parameter is mapped to "a=ptime" in the SDP. The default is 20 ms.
- **maxptime** — Specifies the maximum packetization interval in milliseconds. The default is 100 ms.
- **minptime** — Specifies the minimum packetization interval in milliseconds. The default is 20 ms.
- **maxaveragebitrate** — Specifies the maximum average rate of bits received for a session in bits per second. Bit rates of 5000 to 30000 bps are transcodable by the DSP.
- **usedtx** — Specifies whether the SILK decoder utilizes Discontinuous Transmission (DTX). The possible values are 0 (no) and 1 (yes). The default is 0.

The payload type is dynamic for this codec.

Sample media-profile configuration for adding SILK

Parameter	Value
name	SILK
subname	wideband
media-type	audio
payload-type	103
transport	RTP/AVP
clock-rate	16000
req-bandwidth	0
frames-per-packet	0
parameters	
average-rate-limit	5000
peak-rate-limit	0
max-burst-size	0
sdp-rate-limit-headroom	0
sdp-bandwidth	enabled
police-rate	0
standard-pkt-rate	0

Monitoring and Debugging

CLI commands:

The **show sipd codecs** command is modified to add **SILK Count**.

SNMP:

- New SNMP OID **apSysXCodeSILKWBCapacity** is added to transcoding utilization statistics as reported in the **apSysMgmtGroupTrap**. When utilization falls below 80%, the **apSysMgmtGroupClearTrap** is sent.

- SILK realm statistic **apCodecRealmCountSILK** is added to the **apCodecRealmStatsEntry** table located in the `ap-tc.mib`.

Alarms:

Licensed SILK Transcoding Capacity Threshold Alarm — A warning level alarm that doesn't affect health is triggered when the SILK transcoding utilization exceeds 95% of licensed capacity. The alarm is cleared when the SILK transcoding utilization falls below 80% of licensed capacity.

Configuring Transcoding

Codec Policy Configuration

Transcoding is configured by creating codec policies and referencing them from a realm configuration.

ACLI Configuration Instructions

The parameters that you can configure are name, allow-codecs, add-codecs-on-egress, order-codecs, and ptime. The following section provides brief explanations of how these parameters work, and how you configure each of them.

> **Note:**
>
> A single codec policy can be reused for any number of realms.

To access the configuration parameters for codec policies:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **codec-policy** and press Enter.

   ```
   ORACLE(media-manager)# codec-policy
   ```

   From this point, you can start configuring your codec policy.

Naming Codec Policies

The codec policy's name is important not only because it uniquely identifies the policy, but because it is the name you will enter into your realm configuration's **codec-policy** parameter. It is important to apply the correct policy to the appropriate realm.

To set the codec policy's name:

1. **name**—Set the name for this codec policy, and note it for future reference when you apply codec policies to realms. This parameter is required, and has no default.

   ```
   ORACLE(codec-policy)# name private
   ```

Removing Allowing and Adding Codecs

The Oracle USM removes and allows codecs using the **allow-codecs** parameter. Refer to the Codec Policy Definition section of this chapter for configuration information.

- allow-codecs—The **allow-codecs** parameter takes a list of codecs that you want to pass through the Oracle USM and can explicitly allow them to remain in the SDP for the next step; codecs not matching the items on this list are removed. This parameter is required.
- add-codecs-on-egress—The **add-codecs-on-egress** parameter sets the codecs that the Oracle USM adds to an offer if that codec is not already there. This parameter applies only to the egress policy.
  For allow-codecs, order-codecs, and add codecs to codec policies:

You can configure and edit these two transcoding parameters as ACLI lists, meaning that there are **add** and **delete** commands associated with each. You type the name of the parameter, choose the operation you want to perform on the list (adding or deleting), and then specify the data that you want to add or remove.

The examples in the procedure that follows show you how to add to the lists you are configuring. To remove items from the **allow-codecs** list, simply replace the **add** command you see in these example with **delete** and the items you want to remove.

If you want to overwrite previous values, you can enter the command, a Space and the items in the list enclosed in quotes (“).

1. **allow-codecs**—Enter a list of codecs that are allowed to pass through the Oracle USM. Use the syntax in the Transcodable Codecs section of this chapter. To allow all codecs, enter an asterisk (*).

   ```
   ORACLE(codec-policy)# allow-codecs *
   ```

   When multiple items are added, enclose them in quotes. For example:

   ```
   ORACLE(codec-policy)# allow-codecs G729 G711 AMR
   ```

2. **add-codecs-on-egress**—Enter the codecs that you want added to the SDP offer for the egress codec policy. If you leave this parameter blank, then the Oracle USM will not add codecs to the SDP answer. This parameter cannot be wildcarded; other possible value are listed in the Transcodable Codecs section of this chapter.

   ```
   ORACLE(codec-policy)# add-codecs-on-egress G729
   ```

   If you need to modify the list of configured codecs, you must enter the complete list at once.

Ordering Codecs

Codec policy can specify the order that codecs appear in the SDP offer or answer.

To configure an order which codecs appear in the offer:

1. **order-codecs**—Enter the order in which you want codecs to appear in the SDP offer or answer. You can enter them in any of the ways described in the preceding explanation.

   ```
   ORACLE(codec-policy)# order-codecs G711 * G729
   ```

Transrating Configuration

The following procedure explains how to configure transrating for a codec policy. This codec policy must be applied as an egress codec policy.

To configure forced ptime for a codec policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **network-parameters** and press Enter.

   ```
   ORACLE(media-manager)# codec-policy
   ORACLE(codec-policy)#
   ```

4. If you are adding support for this feature to a pre-existing configuration, then you must select the specific configuration instance, using the ACLI **select** command.

   ```
   ORACLE(codec-policy)# select 1
   ```

   You can now configure forced ptime.

5. **force-ptime**—Set this parameter to **enabled** to enable forced ptime for this codec policy.
6. **packetization-time**—Enter the ptime in ms to use in the realm where this codec policy is active. Valid values are 10, 20, 30, 40, 50, 60, 70, 80, 90 ms. The default value is 20 ms.
7. Save your work using the ACLI **done** command.

Applying a Codec Policy to a Realm

Once you have configured a codec policy, you apply it to a realm by configuration name.

To apply a codec policy to a realm:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. This value is the same as the one you entered in the name parameter for the codec policy you want to use for this realm. There is no default for this parameter.

   ```
   ORACLE(realm-config)# codec-policy private
   ```

Media Profile Configuration

Media profiles must be created and then defined when you want to override the Oracle USM's default media profiles.

ACLI Configuration Instructions and Examples

The parameters that you can configure are name, allow-codecs, add-codecs-on-egress, order-codecs, and ptime. The following section provides brief explanations of how these parameters work, and how you configure each of them.

Creating User-Defined Ptime per Codec

To change the Oracle USM's default ptime for a specific codec, you must create a media profile configuration element. In the **parameter** parameter, you set the ptime to the value of your choosing.

> **Note:**
>
> The frames-per-packet parameter in the media profile configuration element is NOT used for setting a user defined ptime for that codec.

To configure a new ptime value for a codec:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

   If you are adding ptime to a pre-existing media profile, then you must select (using the ACLI select command) the configuration that you want to edit. If you are adding ptime to an undefined media profile, you must create it first.

4. **name**—Type the name of the codec for which you are creating a new default ptime.

   ```
   ORACLE(media-profile)# name pcmu
   ```

5. **payload-type**—Enter the well-known payload type for this codec.

   ```
   ORACLE(media-profile)# payload-type 0
   ```

6. **parameter**—Set the ptime by typing **parameter**, a Space, **ptime=**, the new ptime value. Then press Enter. For example:

   ```
   ORACLE(media-profile)# parameter ptime=40
   ```

7. Save your work using the ACLI **done** command.

Media Type Subnames

You can define multiple versions of a media profile for a single codec by using the subnames feature. You can then reference the new media profile by a combination of the media profile name and media profile subname.

Some media types are not unique per just their value in an SDP m= line, they must be uniquely identified by looking at additional SDP parameters. For example, you can define a media profile for G729, when only the parameter and value **annexb=yes** is present in the SDP. By creating a media profile + subname that defines both a media type and parameter, you can perform various operations on G729 only when **annexb=yes** is encountered.

Some applications of media type subnames are:

- maintaining different versions of the same codec with different bandwidth ceilings
- maintaining different versions of the same codec with different ptimes
- grouping codecs by using customer as a subname
- grouping codecs by using realm as a subname

SDP Parameter Matching

This feature matches parameters in the **a=fmtp**, codec-specific SDP **a=** line. It does not try to match a global **m=** line attribute like **a=ptime**.

Using Subnames with Codec Policies

Media profiles are defined and referenced in the ACLI by a name and subname in the following format

```
<name>::<subname>
```

If no subname has been created for a media profile, you may continue using the media profile name without any subname specifier.

For example, to remove a media profile and subname configured as PCMU::customer1 from all SDP entering the egress realm, you would configure the codec policy **allow-codecs** parameter as follows:

```
allow-codecs PCMU::customer1:no
```

media-profile subtype Configuration Restrictions

media-profiles are subject to stringent configuration restrictions. You must avoid creating a **media-profile** with configured **subtype** parameter that does not substantively differ (in all additional parameters) from the default (unconfigured) media profile. An example of an invalid configuration is **media-profile** > **name** of g729, and a **media-profile** > **subname** of g729, with no additional parameter configurations other than the default values. Such configuration can cause unexpected behaviors and must be avoided.

Subname Syntax and Wildcarding

You can wildcard one or both portions (name and subname) of a media type and subname pair:

- When you wildcard the **name** portion of the value, you can provide a specific subname that the Oracle USM uses to find matching media profiles.
- When you wildcard the subname portion of the value, you can provide a specific **name** that the Oracle USM uses to find matching media profiles.

The following table defines and explains subname wildcarding and syntax:

Syntax	Example Value	Description
<name>	PCMU	Matches any and all media profiles with the name value configured as PCMU. This entry has the same meaning as a value with this syntax: <name>::*.
<name>::	PCMU::	Matches a media profile with the name with the name value configured as PCMU with an empty subname parameter.
<name>::*	PCMU::*	Matches any and all media profiles with the name value configured as PCMU with any and all subname configured.
<name>::<subname>	PCMU::64k	Matches a media profiles with the name with the name value configured as PCMU with the subname parameter set to 64k.
*	*	Matches anything, but does not have to be a defined media profile.
::	*::*	Matches any and all media profiles, but requires the presence of media profile configurations.
*::<subname>	*::64k	Matches all media profiles with this subname. You might have a group of media profiles with different names, but the same subname value.
*::	*::	Matches any media profiles with an empty subname parameter.
::	::	Invalid
::*	::*	Invalid

Wildcarding add-codecs-on-egress

It is important to note that you may not configure **add-codecs-on-egress** with a wildcarded subname in a codec policy. You may only add a specific instance of a media type.

Valid:

```
add-codecs-on-egress PCMU
add-codecs-on-egress PCMU::customer1
```

Invalid:

```
add-codecs-on-egress PCMU::*
```

Media Type and Subname Configuration

To use media type subnames with a codec policy, you must first configure a media profile and subname. Then you can configure a codec policy with a media type and subname pair for your application

To use configure a media type and subname:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ORACLE(session-router)#
   ```

3. Type **media-profile** and press Enter.

   ```
   ORACLE(session-router)# media-profile
   ORACLE(media-profile)#
   ```

4. **name**—Type the name of the codec for which you are creating a new default ptime.

   ```
   ORACLE(media-profile)# name g729
   ```

5. **subname**—Enter a description for the use of this subname

   ```
   ORACLE(media-profile)# subname annexb-yes
   ```

   You may now configure this subname's unique attributes. PCMU is created with ptime of 30 in this example.

6. **parameters**—Set the ptime by typing **parameter**, a Space, **ptime=**, the new ptime value. Then press Enter. For example:

   ```
   ORACLE(media-profile)# parameter annexb=yes
   ```

   > **Note:**
   >
   > Remember to configure all additional, required media profile parameters, or they will inherit default values.

7. Save your work using the ACLI **done** command.

Codec Policy Configuration with a Media Type with a Subname

To configure a codec policy with a media type with subname:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **codec-policy** and press Enter.

   ```
   ORACLE(media-manager)# codec-policy
   ```

4. Use the ACLI **select** command to select a codec policy.

   ```
   ORACLE(codec-policy)# select 1
   ```

   You may now enter a media profile with subname to any parameter in the codec policy that accepts a media profile.

5. **allow-codecs**—Enter a list of codecs that this codec policy allows or denies from passing through the Oracle USM. To allow all codecs, enter an asterisk (*).

   ```
   ORACLEcodec-policy# allow-codecs g729::annexb-yes:no
   ```

6. Save and activate your configuration.

Codec and Conditional Codec Policies for SIP

The Oracle USM has the ability to add, strip, and reorder codecs for SIP sessions. This builds on the Oracle USM's pre-existing abilities to route by codec and re-order one codec in an SDP offer by allowing you to configure the order of multiple codecs and to remove specific codecs within the media descriptions in SDP offers.

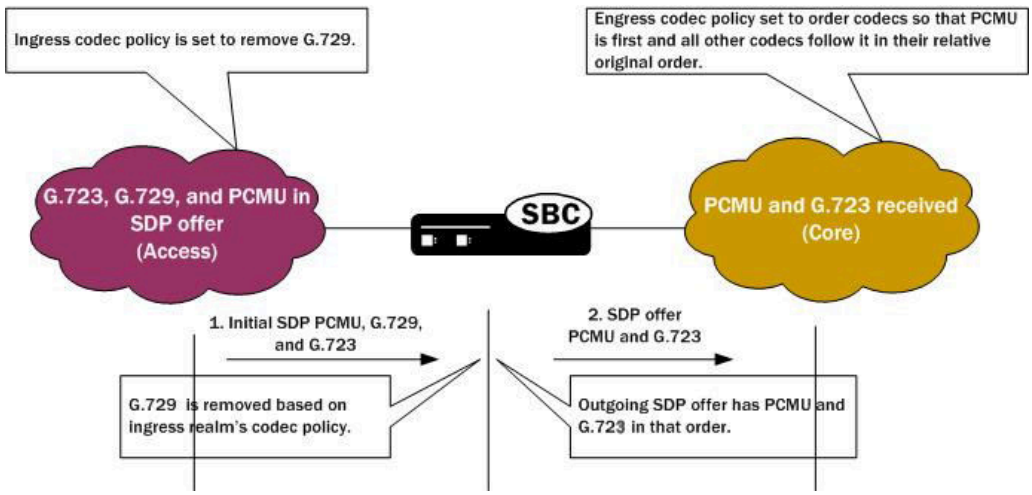
You can enable the Oracle USM to perform these operations on SDP offers by configuring codec policies. Codec policies are sets of rules that specify the manipulations to be performed on SDP offers. They are applied on an ingress and egress basis using the realm and session agent configurations.

Oracle USM supports three types of codec policies:

- Ingress policy—Codec policy that the Oracle USM applies to the SDP offer for incoming traffic
- Egress policy—Codec policy that the Oracle USM applies to the SDP offer for traffic leaving the Oracle USM
- Conditional policy—Codec policy that the Oracle USM applies to the SDP offer for traffic leaving the Oracle USM. A conditional policy differs from an egress policy in providing the capability to perform standard codec manipulations (add, strip and re-order) dynamically, based on the codec list and associated parameters contained in the original SDP offer. Refer to Conditional Codec Policies for specific details regarding the use and construction of these policies.

The Oracle USM applies codec policies during the offer phase of media format negotiation. If codec manipulation is enabled, then the Oracle USM performs the modification according to the specific policy and forwards on the traffic.

For example, when the Oracle USM receives a SIP INVITE with SDP, it refers to the realm through which the INVITE arrived and performs any manipulations specified by an ingress codec policy that may have been assigned to the ingress realm. With the media description possibly changed according to the ingress codec policy, the Oracle USM passes the SDP offer to the outgoing realm so that the an egress codec policy can be applied. Note that the SDP to be evaluated by the egress codec policy may match the original SDP, or it may have been changed during transit through the ingress realm. After applying the egress coded policy, the Oracle USM forwards the INVITE.



Since the offer-answer exchange can occur at different stages of SIP messaging, the assigned ingress and egress roles follow the media direction rather than the signaling direction. It might be, for example, that the offer is in an OK that the Oracle USM modifies.

You can apply codec policies to realms and to session agents; codec policies configured in session agents take precedence over those applied to realms. However, it is not required that there be both an ingress and an egress policy either for realms or for session agents. If either one is unspecified, then no modifications take place on that side. If neither ingress nor egress policies specified, SDP offers are forwarded as received.

Relationship to Media Profiles

For each codec that you specify in a codec policy, there must be a corresponding media profile configuration on the Oracle USM. You configure media profiles in the ACLI via the session-router path. In them, you can specify codec type, transport protocol, required bandwidth, and a number of constraints.

Manipulation Modes

You can configure a codec policy to perform several different kinds of manipulations:

- Allow—List of codecs that are allowed for a certain codec policy; if a codec does not appear on this list, then the Oracle USM removes it. You can wildcard this list with an asterisk (*) so that all codecs are allowed. Further, you can create exceptions to a wildcarded allow list.
  - You make an exception to the wildcarded list of codecs by entering the codec(s) that are not allowed with a **no** attribute. This tells the Oracle USM to allow all codecs except the one(s) you specify.

    ```
    ORACLE(codec-policy)# allow-codecs (* PCMA:no)
    ```

  - You can also create exceptions to allow lists such that audio or video codecs are removed. However, when the allow list specifies the removal of all audio codecs and an INVITE arrives at the Oracle USM with only audio codecs, the Oracle USM behaves in accordance with RFC 3264. This means that the resulting SDP will contain one attribute line, with the media port for the media line set to 0. The terminating side will need to supply new SDP in its reply because the result of the manipulation is the same as an INVITE with no body.

    ```
    ORACLE(codec-policy)# allow-codecs (* audio:no)
    ```

- Order—List of the codecs where you specify their preferred order in the outgoing media offer. The Oracle USM arranges matching codecs according to the rule you set, and any remaining ones are added to the list in the same relative order they took in the incoming media offer. If your list specifies a codec that is not present, then the ordering proceeds as specified but skips the missing codec.
  You can use an asterisk (*) as a wildcard in this list, too. The placement of the asterisk is key, as you can see in the following examples:
  - For an order rule set this way

    ```
    ORACLE(codec-policy)# order (A B C *)
    ```

    codecs A, B, and C will be placed at the front of the codec list in the order specified; all other codecs in the offer will follow A, B, and C in the same relative order they had in the original SDP offer.

  - For an order rule set this way:

    ```
    ORACLE(codec-policy)# order (* A B C)
    ```

    codecs A, B, and C will be placed at the end of the codec list in the order specified; all other codecs in the offer will come before A, B, and C in the same relative order they had in the original SDP offer.

  - For an order rule set this way

    ```
    ORACLE(codec-policy)# order (A * B C)
    ```

    codec A will be placed at the beginning of the codec list, to be followed by all other codecs in the offer in the same relative order they had in the original SDP offer, and then B and C will end the list.

- Force—An attribute you can use in the allow list with one codec to specify that all other codecs should be stripped from the outgoing offer. You can specify multiple forced codecs in your rules.
  - If you set multiple codecs in the allow list and one of them is forced, then the outgoing offer will contain the forced codec.
  - If you set multiple codecs in the allow list and the one that is forced is not present in the offer, then the Oracle USM will select a non-forced codec for the outgoing offer.

    ```
    ORACLE(codec-policy)# allow (PCMU G729:force)
    ```

    You cannot use the force attribute with a wildcarded allow list.
- No—An attribute that allows you to strip specified codecs or codec types from a wildcarded allow list.

  ```
  ORACLE(codec-policy)# allow (* PCMA:no)
  ```

In-Realm Codec Manipulation

In addition to being able to apply codec policies in realms, the realm configuration supports a setting for determining whether codec manipulation should be applied to sessions between endpoints in the same realm.

In-realm codec manipulation can be used for simple call flows that traverse two realms. If the originating and terminating realms are the same, the Oracle USM checks to see if you have enabled this capability. If you have enabled it, then the Oracle USM performs the specified manipulations. If this capability is not enabled, or if the realm's media management in realm (**mm-in-realm**) setting is disabled, then the Oracle USM does not perform codec manipulations.

For more complex calls scenarios that involve call agent or reinitiation of a call back to the same realm, the Oracle USM does not perform in-realm codec manipulation.

Conditional Codec Policies

A codec policy performs actions conditionally when any of its parameters includes a conditional value. A conditional value includes a target codec paired with a requirement for executing the action. The Oracle USM manipulates SDP according to this value pair if the ingress SDP or a previous manipulation to the SDP meets the condition's criteria. The user configures this "conditional manipulation" by extending upon the syntax of three core parameters in the codec-policy configuration element, including:

- allow-codecs,
- add-codecs-on-egress, and
- order-codecs.

The system establishes conditions on a codec policy as a sequence of allowing, adding, and re-ordering. Each step in this sequence can occur with or without conditions. Allowing is required. Any applied policy, whether or not it is conditional, without an allow blocks all traffic. Allow all, using the wildcard asterisk character is a typical setting. Adding only applies to egress policies.

To establish conditions, the user configure the parameter with pairs that consist of target codecs followed by the condition(s) that trigger the action. Each policy parameter can include one or more of these pairs. When configuring a parameter with multiple values, the user encloses them within parenthesis, whether or not there are conditions.

The system processes all policies serially, regardless of whether any includes a condition. Specifically, the system first determines which codecs to allow, then which to add (none on ingress), then the codec order. The system also processes parameter values serially. The user, therefore, must configure all parameter values based on what may have been changed previously. This is particularly important when using both ingress and egress codec policies. Consistent with this serial concept, egress policies operate on what the system presents to them, which includes the results of any ingress policies.

Note that the use of conditional **order-codecs** is often done in conjunction with **add-codecs-on-egress** to define the location of user-added codecs to the list presented at egress. When **order-codecs** is not used, the system places all added codecs at the beginning of the list in the order they were added. It is often desirable to place an added codec in a different position, using **order-codecs**.

Conditional Codec Lists

Conditional codec policies are constructed using existing ACLI configuration commands — **allow-codecs**, **add-codecs-on-egress**, and **order-codecs** — in conjunction with keywords and operators. Conditions are defined by a continuous character string (no spaces allowed) that starts with the **:(** character sequence and is terminated by a closing parenthesis. For example,

```
ORACLE(codec-policy)# allow-codecs PCMU:(!G729)
```

which can be interpreted as — allow PCMU if the G729 codec is not in the SDP offer after ingress codec policy processing.

An example of using **add-codecs-on-egress** is:

```
ORACLE(codec-policy)# add-codecs-on-egress PCMU:(PCMA)
```

which can be interpreted as — add PCMU if PCMA codec is in the SDP offer after ingress codec policy processing.

If PCMA is in the SDP offer after ingress codec policy processing, the **add-codecs-on-egress** ACLI command is treated as **add-codecs-on-egress PCMU**. If PCMA is not in the SDP offer after ingress codec policy processing, **add-codecs-on-egress** is treated as empty.

Both the conditioned codec and/or the condition itself can contain subnames. For example,

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::TEST0)
```

which can be interpreted as — add AMR::ONE if AMR::TEST0 codec is in the SDP offer after ingress codec policy processing.

Codecs contained in the condition can be wildcarded. For example,

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*)
```

which can be interpreted as — add AMR::ONE if any AMR codec is in the offer after ingress codec policy processing.

An example of using **order-codecs** is:

```
ORACLE(codec-policy)# order-codecs (PCMU:(PCMU) *)
```

which can be interpreted as — set the codec order to PCMU followed by the list after ingress codec policy processing if PCMU is not present. When the system adds a codec, the codec goes to the end of the offered list. This conditional format may be used to place an added codec at the front of list.

Conditional Codec Operators

Three logical operators are available to construct conditional lists

the OR operator (|)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*|PCMU)
```

which can be interpreted as — add AMR::ONE if any AMR:: codec is in the SDP offer after ingress codec policy processing, or if PCMU is in the SDP offer after ingress codec policy processing.

the AND operator (&)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(AMR::*&PCMU)
```

which can be interpreted as — add AMR::ONE if any AMR:: codec is in the SDP offer after ingress codec policy processing, and if PCMU is in the SDP offer after ingress codec policy processing.

the NOT operator (!)

```
ORACLE(codec-policy)# add-codecs-on-egress AMR::ONE:(!AMR::*)
```

which can be interpreted as — “add AMR::ONE if no AMR:: codec is in the SDP offer after ingress codec policy processing.

Each operator applies only to the codec immediately following it. Operators are processed left to right until all conditions have been tested. The condition result is accumulated as each of the conditions is processed. For example:

```
ORACLE(codec-policy)#add-codecs-on-egress AMR::ONE:
(!AMR::TEST1&AMR::TEST0|AMR::TEST2)
```

- Test SDP offer for AMR::TEST1 (a NOT operation).
  If AMR:TEST1 is NOT present, set accumulated result to TRUE.

  If AMR:TEST1 is present, set accumulated result to FALSE.
- Test SDP offer for AMR::TEST0 (an AND operation).
  If AMR is present, no change to accumulated result.

  If AMR is not present, set accumulated result to FALSE.
- Test SDP offer for AMR::TEST2 (an OR operation).
  If AMR is present, set accumulated result to TRUE.

If AMR is not present, no change to accumulated result.

Multiple conditions can be concatenated; in this case, individual conditions are separated by SPACE characters and the ACLI command argument is bracketed with double quotation marks ...). For example:

```
ORACLE(codec-policy)#add-codecs-on-egress (PCMU G729:(G726) G723:(PCMA))
```

- PCMU is unconditionally added to the egress codec list.
- Process the first condition — G729:(G726)
  If G726 is present, the result is TRUE; add G729 to the egress codec list.

  If G726 is not present, the result is FALSE; do not add G729 to the egress codec list.
- Process the second condition — G723:(PCMA)
  If PCMA is present, the result is TRUE; add G723 to the egress codec list.

  If PCMA is not present, the result is FALSE; do not add G723 to the egress codec list.

ACLI Instructions and Examples

This section gives instructions and examples for how to configure codec policies and then apply them to realms and session agents. It also shows you how to configure settings for in-realm codec manipulation.

Creating a Codec Policy

To create a codec policy:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the signaling-related configurations.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type codec-policy and then press Enter.

   ```
   ORACLE(media-manager)# codec-policy
   ORACLE(codec-policy)#
   ```

4. **name**—Enter the unique name for the codec policy. This is the value you will use to refer to this codec policy when you apply it to realms or session agents. This parameter is required and is empty by default.

5. **allow-codecs**—Enter the list of media format types (codecs) to allow for this codec policy. In your entries, you can use the asterisk (*) as a wildcard, the force attribute, or the no attribute so that the allow list you enter directly reflects your configuration needs. Enclose entries of multiple values in parentheses (( )). For more information, refer to the Manipulation Modes section above.

   The codecs that you enter here must have corresponding media profile configurations.

   allow-codecs can be used to construct ingress, egress, or conditional codec policies. For details of conditional codec policies, refer to Conditional Codec Policies.

6. **add-codecs-on-egress**—Enter the codecs that the Oracle USM adds to an egress SDP offer if that codec is not already there. This parameter applies only to egress offers. For more information, refer to the Manipulation Modes section above.

The codecs that you enter here must have corresponding media profile configurations.

add-codecs-on-egress can be used to construct ingress, egress, or conditional codec policies. For details of conditional codec policies, refer to Conditional Codec Policies.

7. **order-codecs**—Enter the order in which you want codecs to appear in the outgoing SDP offer. Remember that you can use the asterisk (*) as a wildcard in different positions of the order to directly reflect your configuration needs. Enclose entries of multiple values in parentheses (( )). For more information, refer to the Manipulation Modes section above.

   The codecs that you enter here must have corresponding media profile configurations.

   order-codecs can be used to construct ingress, egress, or conditional codec policies. For details of conditional codec policies, refer to Conditional Codec Policies.

8. Save and activate your configuration.

   Your codec policy configuration will resemble the following example:

```
codec-policy
        name            private
        allow-codecs    g723:no pcmu video:no
        order-codecs    pcmu *
```

Applying a Codec Policy to a Realm

Note that codec policies defined for session agents always take precedence over those defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **media-manager** and press Enter.

```
ORACLE(configure)# media-manager
```

3. Type **realm-config** and press Enter.

```
ORACLE(media-manager)# realm-config
```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

Applying a Codec Policy to a Session Agent

Note that codec policies that are defined for session agents always take precedence over those that are defined for realms.

To apply a codec policy to a realm:

1. In Superuser mode, type **configure terminal** and press Enter.

```
ORACLE# configure terminal
```

2. Type **session-router** and press Enter.

```
ORACLE(configure)# session-router
```

3. Type **session-agent** and press Enter.

   ```
   ORACLE(session-router)# session-agent
   ```

   If you are adding support for this feature to a pre-existing session agent, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-policy**—Enter the name of the codec policy that you want to apply to this realm. By default, this parameter is empty.
5. Save and activate your configuration.

In-Realm Codec Manipulations

To enable in-realm codec manipulations:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter.

   ```
   ORACLE(configure)# media-manager
   ```

3. Type **realm-config** and press Enter.

   ```
   ORACLE(media-manager)# realm-config
   ```

   If you are adding support for this feature to a pre-existing realm, then you must select (using the ACLI **select** command) the realm that you want to edit.

4. **codec-manip-in-realm**—Enter the name of the codec policy that you want to apply to this realm. The default value is **disabled**. The valid values are:
   - enabled | disabled
5. Save and activate your configuration.

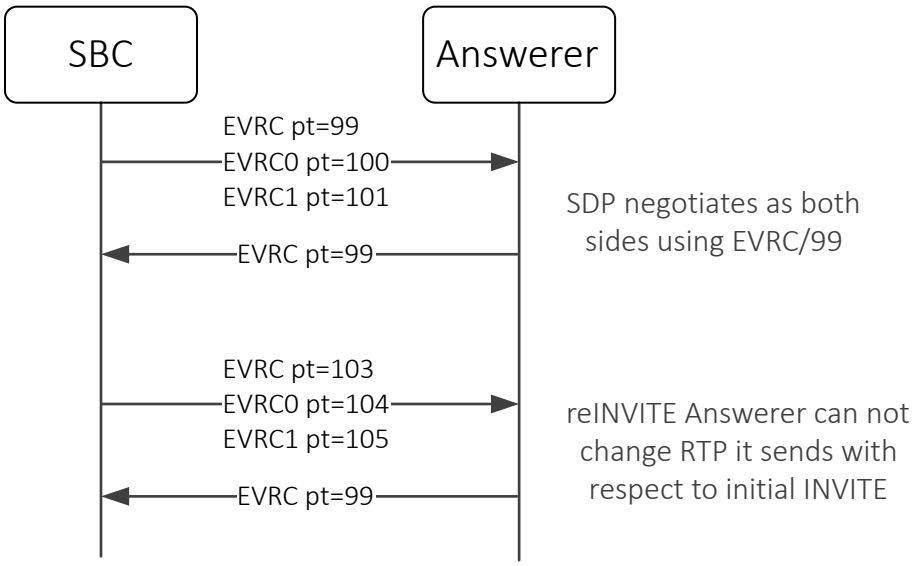
Asymmetric Dynamic Payload Types Enablement

Transcoding Support for Asymmetric Dynamic Payload Types supports calls with asymmetric payload types such that a codec offered with one payload type and answered with another payload type is acceptable to the Oracle USM. This behavior requires transcoding resources.

The Oracle USM's default behavior when an endpoint answers an SDP offer with a different payload type than offered is to use what the endpoint replied with as if that were what the Oracle USM included in its SDP offer. The Oracle USM would then expect to receive the same payload type that the endpoint offered. This is referred to as symmetric payload type mapping, whereby the payload type number is the same for both media flows.

The Oracle USM needs to support the asymmetric case when codecs with dynamic payload types are used. For example, a call may be set up with the dynamic codec using one payload type value, and a reINVITE may use a different payload type value for the same codec. Because of the range of remote UE equipment's behavior in a reINVITE case, the Oracle USM can support this via its Asymmetric Dynamic Payload Type feature. That is, some devices do not necessarily use the currently negotiated payload type, they may use previously negotiated payload types. As such devices interact with the default Oracle USM behavior, one-way audio

may result.



For transcoded calls, enabling this feature places no additional load on the system. But for calls that are not transcoded, this feature consumes transcoding resources.

Configure Transcoding for Asymmetric Dynamic Payload Types

Transcoding support for asymmetric dynamic payload types enables the Oracle USM to perform transcoding when the Real-time Transport Protocol (RTP) is offered with one payload type and is answered with another payload type. Enable transcoding for asymmetric dynamic payload types from the command line.

Before You Begin

- Confirm that you are in Superuser mode.

Procedure

1. Access the **media-manager-config** configuration element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

2. Type select to begin editing.

   ```
   ORACLE(media-manager-config)# select

   ORACLE(media-manager-config)#
   ```

3. Use the **options +audio-allow-asymmetric-pt** command to enable support for asymmetric payload types.

   ```
   ACMEPACKET#(media-manager) options +audio-allow-asymmetric-pt
   ACMEPACKET#(media-manager)
   ```

4. Type **done** to save your configuration.

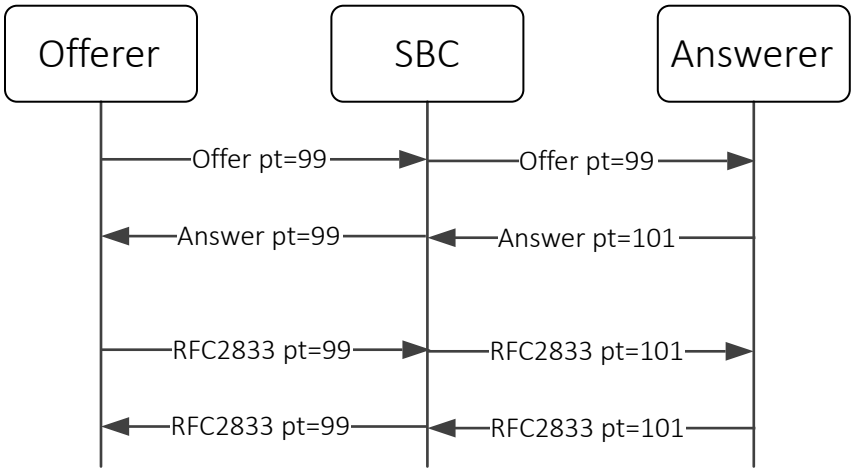
Asymmetric Payload Type Support for RFC2833 Interworking

RFC3264 describes "asymmetric" behavior when each participant of a call leg sends can expect a different payload-type value for the RTP stream. Conversely, when the two participants in a call leg must send and receive an RTP stream using the same Payload Type value, the behavior is symmetric. The Oracle USM can be configured to behave in an RFC-compliant manner, which is to support the asymmetric case. Symmetric-only cases are enforced by default.

Default Symmetric RFC2833 Interworking

The Oracle USM's default behavior when an endpoint answers an SDP offer with a different payload type value for RFC 2833 telephone-events than what the SBC sent, is to mimic what the Answerer replied with as the payload type value it (the SBC) will expect. This is referred to as symmetric payload type mapping, whereby the SBC sends RTP with the payload type value that the answerer replied with in the signaling phase of the call, and also expects that same payload type value when RTP is sent to it (despite having sent a different value in the initial SDP offer).

The following example shows the default behavior. When the Answerer returns PT 101, the Oracle USM is prepared to enforce symmetric-only behavior on the egress realm by expecting RFC2833 packets with Payload Type 101, even though it offered Payload Type 99.

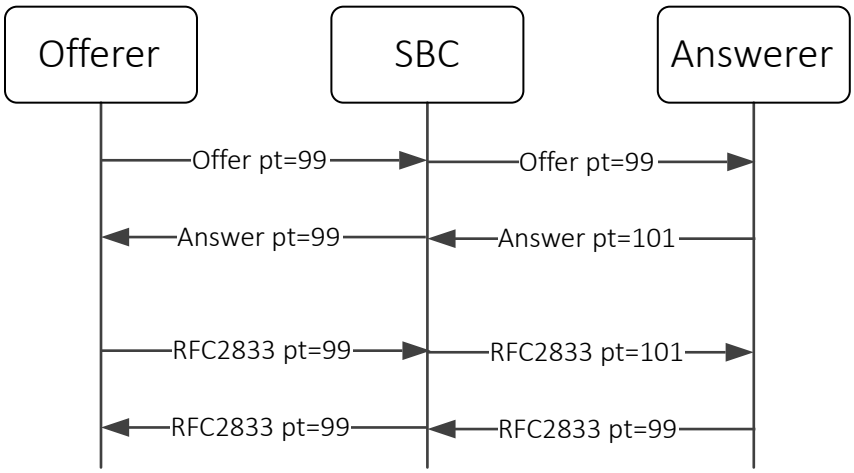


Since ingress-side behavior is to reply with the payload type offered, it expects payload type 99 and sends payload type 99. Thus to facilitate RFC2833 interworking, the payload types will be remapped from 99 -> 101 in the eastbound direction and 101 -> 99 in the westbound direction.

Asymmetric RFC2833 Interworking

The Oracle USM supports the asymmetric, RFC-compliant case by configuration. In this case, as the signaling is set up on the egress side of the call, the Oracle USM indicates it expects to receive RFC2833 packets with Payload Type 99, while the answerer indicates it expects to receive RFC 2833 packets with Payload Type 101. This valid call state can be accommodated

by the SBC when the **rfc2833-allow-asymmetric-pt** option is configured.



Therefore, the Oracle USM being able to support asymmetric payload types for RFC2833 packets, will remap from 99 ->101 in the eastbound direction and not have to remap the payload type value in the westbound direction.

The **rfc2833-allow-asymmetric-pt** option can be configured on a **sip-interface** or **session-agent** where this behavior is desired.

RFC2833 Interworking With and Without Transcoding Resources

In addition to the above behavior, it is important to note that the Oracle USM acts differently for forwarding media streams depending on if the call is transcoded or not. If the call is not transcoded, any RTP stream, regardless of the payload type being different from what was negotiated in the signaling phase will be forwarded through the system as is. In either of the above examples, if the Answerer sends RFC2833 packets with payload type 105, that RTP stream will be forwarded to the ingress leg of the call, untouched by payload mapping. This behavior could potentially mask the fact that the Oracle USM's egress interface is expecting a different payload type value than what is sent since the traffic is still forwarded to the ingress realm.

If the call is transcoded via **codec-policy**, any RTP stream that uses an unexpected payload type (as different from what was negotiated in the signaling phase) will be dropped by the system. In the above example, if the Answerer sends RFC2833 packets with payload type 105, that RTP stream will be dropped.

DTMF Indication over HD Audio Codecs

When performing DTMF transcoding while HD Audio codecs are present, Oracle USM accounts for telephone-event tone indication at clock rates that match those of the HD audio codecs.

The telephone-event tone indication's clock rate, when sent alongside an HD codec, must match the audio codec's clock rate. While many non-HD codecs use 8000 Hz clock rates, HD codecs can use clock rates of 16000 Hz. Transcoding processing takes these two telephone-event clock rates into account when performing SDP manipulation. If the wrong clock rate is used, RFC2833 telephone-event indications may be relayed incorrectly.

On the ingress side of the call, if a telephone-event is received, and no audio codec with a matching clock rate is received, the ingress-side SDP response will have the unmatched telephone-event removed.

If the **allow-codec** parameter uses the :no tag to remove the last of an audio codec that matches a telephone-event (8000 or 16000) from SDP, then the telephone-event of that clock rate (if present) will be removed from the SDP too.

When codec policy dictates to add AMR-WB, and the received SDP contains PCMU/PCMA and telephone-event (8000), the SDP offer sent from the egress interface will include telephone-event (16000)

If **rfc2833-mode** is set to **preferred**, the Oracle USM adds telephone-event 16000 to outbound SDP if AMR-WB is present in the outbound SDP.

On the egress side of the call if the SDP contains a telephone-event without an audio codec with a matching clock rate, an appropriate audio codec will be added. If codec policy adds a telephone-event, then the SBC analyzes the audio codecs in the outbound SDP and ensures that matching telephone events (8000 and/or 16000) are present.

Once the Oracle USM determines the SDP to forward, the order of telephone-event clock rates will be modified to match the order of audio codec rates. Thus if AMR-WB is the top codec followed by codecs with 8000 Hz clock rates, the telephone-event with a clock rate of 16000 Hz will be listed above telephone-events with 8000 Hz clock rates.

RFC2833 and KPML Interworking

Keypad Markup Language (KPML) is used to indicate DTMF tones in SIP messaging. KPML is used by the Key Press Stimulus Package as a SIP Event Notification Package. Endpoints will send DTMF tone indications via NOTIFY messages. The Oracle USM can engage in the Event Notification with an endpoint on one call leg and perform digit translation to RFC 2833 telephone-events on the other call leg.

KPML to RFC2833

KPML to RFC2833 interworking requires that the INVITE request or response's SDP does not contain telephone-event, and is received from

- A session agent with **kpml-2833-interworking** set to enabled
- A session agent with **kpml-2833-interworking** set to inherited from the SIP interface (with **kpml-2833-interworking** set to enabled)
- A previous hop that is not a session agent, and the SIP interface the message received on has **kpml-2833-interworking** set to enabled.

The egress side of the call must have **rfc2833-mode** set to **preferred** in either the SIP interface or session agent, so that the Oracle USM inserts telephone-event in the SDP. Setting **rfc2833-mode** to dual is unsupported. The answerer must respond to the invite accepting of telephone-event media. The Allow-Event: kpml header is removed from the INVITE request or response when KPML-interworking is not set to enabled on the next hop.

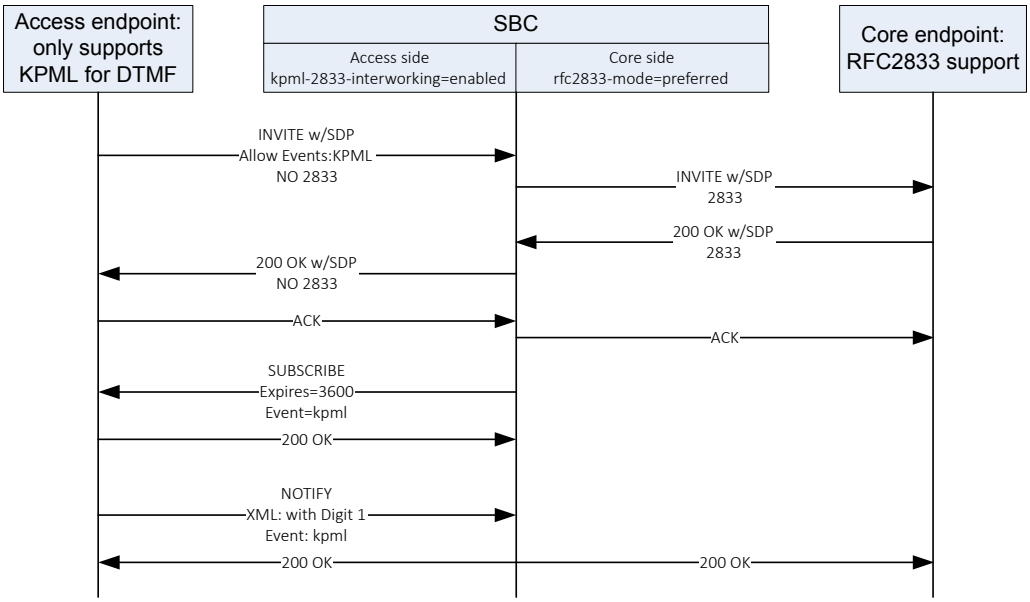
If the INVITE succeeds, the Oracle USM replies with a SUBSCRIBE request for the KPML event to the caller.

If the caller replies to the SUBSCRIBE with a 2xx, all subsequent NOTIFY request received on the dialog are processed and replied to with a 200 OK. Each digit in the NOTIFY request will generate a telephone-event on the egress side of the call.

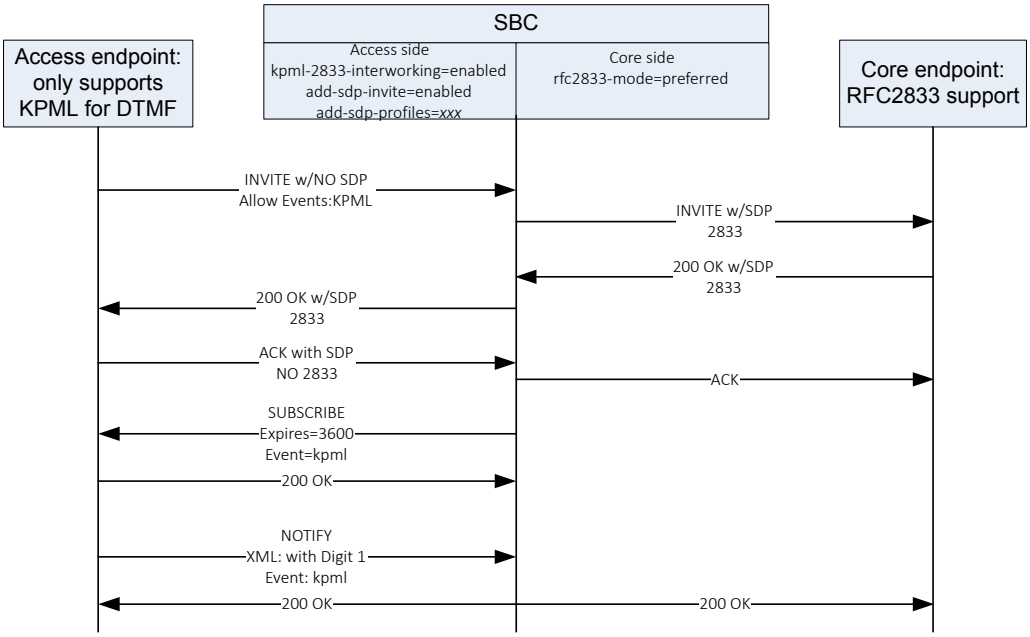
The Oracle USM generates a refresh SUBSCRIBE before the subscription expires.

If the call negotiates to RFC 2833 to RFC 2833, no KPML interworking will be performed.

The following diagram shows the standard case where KPML to RFC 2833 interworking is performed.



KPML to RFC 2833 translation is also supported when used in conjunction with SDP insertion on the KPML side of the call. For example:



RFC 2833 to KPML

RFC2833 to KPML interworking requires that the INVITE request or response's does not contain Allow-Event: kpml. When sending an INVITE when RFC2833 interworking applies, an Allow-Event: kpml header is added to the INVITE when the message's next hop on egress is either :

- A session agent with **kpml-2833-interworking** set to enabled
- A session agent with **kpml-2833-interworking** set to inherited from the SIP interface (with **kpml-2833-interworking** set to enabled)

- Not a session agent, and the SIP interface the message is sent from has **kpml-2833-interworking** set to enabled.

If the scenario passes the above test and the Allow-Event: kpml header is added to the INVITE:

When the Oracle USM subsequently receives a SUBSCRIBE request within the INVITE created dialog for the kpml event, it will accept the subscription and respond with a 200 OK. At this point, the Oracle USM can generate a KPML NOTIFY request with a KPML digit corresponding to each 2833 telephone-event received from the far end until

- the INVITE dialog is terminated due to a BYE
- the subscription expires
- the subscription is terminated with a subscribe request Expires: 0

The following diagram shows a typical RFC2833 to KPML interworking call flow. Note the following:

- While the example has SDP in the INVITE, delayed offer scenarios where the SDP exchange occurs in the 20 0OK and the ACK are supported
- The **rfc2833-mode** parameter in the in either the SIP interface or session agent, is considered in the interworking. See next section:
- If the call negotiates to RFC 2833 to RFC 2833, no KPML interworking will be performed.
- In RFC 2833 to KPML scenarios, if the SUBSCRIBE is not received quickly enough, RFC 2833 digits will be dropped.

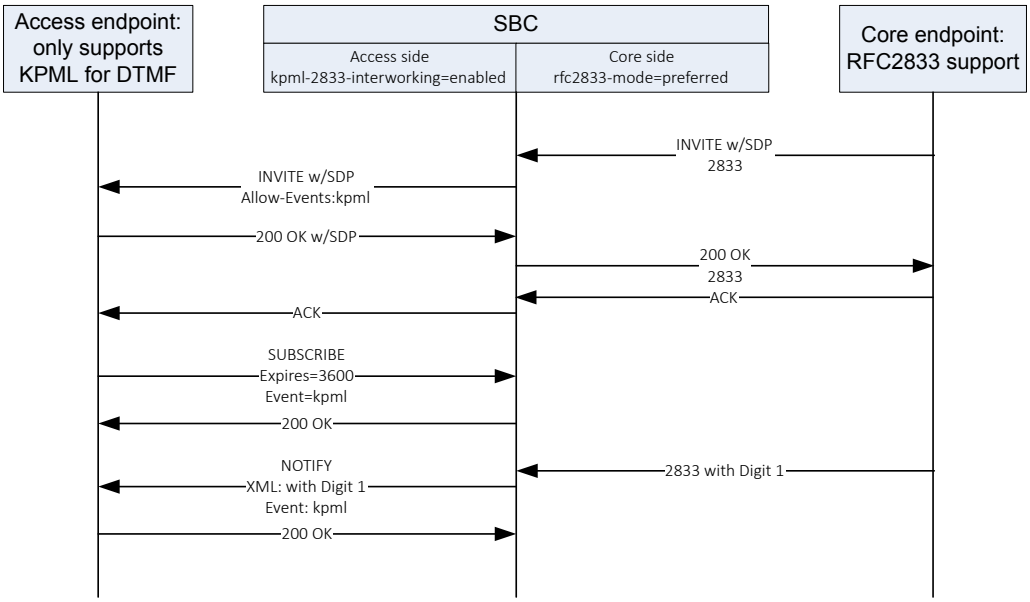


Table 19-1 RFC2833 to KPML interaction with rfc2833-mode

Originator	Terminator	SBC Behavior
rfc2833-mode=transparent, offers 2833 SDP	rfc2833-mode=transparent, answers NO 2833 SDP	Forwards 2833 offer, Adds Allow-Event: kpml, passes non-2833 capability to Originator (no 2833 support)

Table 19-1 (Cont.) RFC2833 to KPML interaction with rfc2833-mode

Originator	Terminator	SBC Behavior
rfc2833-mode=preferred, offers 2833 SDP	rfc2833-mode=transparent, answers NO 2833 SDP	Forwards 2833 offer, Adds Allow-Event: kpml, inserts 2833 capability to Originator, performs 2833 to KPML translation
rfc2833-mode=transparent, offers 2833 SDP	rfc2833-mode=preferred, answers NO 2833 SDP	Forwards 2833 offer, Adds Allow-Event: kpml, passes non-2833 capability to Originator (no 2833 support)
rfc2833-mode=preferred, offers 2833 SDP	rfc2833-mode=preferred, answers NO 2833 SDP	Forwards 2833 offer, Adds Allow-Event: kpml, inserts 2833 capability to Originator, performs 2833 to KPML translation

KPML-2833 Interworking on a SIP Interface Configuration

To configure KPML - 2833 interworking on a SIP interface:

1. Access the **sip-interface** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-interface
ORACLE(sip-interface)#
```

2. Select the **sip-interface** object to edit.

```
ORACLE(sip-interface)# select
<RealmID>:
1: realm01 172.172.30.31:5060

selection: 1
ORACLE(sip-interface)#
```

3. **kpml-interworking**—Set this parameter to enabled to use KPML-2833 interworking.
4. Type **done** to save your configuration.

KPML-2833 Interworking on a Session Agent Configuration

To configure KPML - 2833 interworking on a session agent:

Enter the prerequisites here (optional).

Enter the context of your task here (optional).

1. Access the **session-agent** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-agent
ORACLE(session-agent)
```

2. Select the **session-agent** object to edit.

```
ORACLE(session-agent)# select
<hostname>:
1: 192.168.100.101:1813
```

```
selection: 1
ORACLE(session-agent)#
```

3. **kpml-interworking**—Set this parameter to enabled for the Oracle USM to interwork RFC2833 from the other call leg to the call leg running through this session agent. This parameter may be set to inherit to use the **kpml-interworking** value configured on the receiving SIP interface.
4. Type **done** to save your configuration.

Maintenance and Troubleshooting

show mbcd errors

The **show mbcd errors** command displays statistics related to MBCD task errors. The following fields are explained:

- XCode Internal Errors—Number of uncategorized errors due to Transcoding session error.
- XCode Alloc Errors—Number of times that buffer allocation failed for transcoding tasks.
- XCode Update Errors—Number of errors encountered when attempting to update an entry in the Transcoding table upon receipt of the first packet for a media flow.
- XCode Delete Errors—Number of errors encountered when attempting to delete an entry in the Transcoding table.
- XCode Over Cap Errors—Number of Transcoding sessions denied once session capacity is reached.
- XCode Over License Cap—Number of Transcoding sessions denied once license capacity is reached.

```
ORACLE# show mbcd errors
13:22:50-126
MBC Errors/Events              ---- Lifetime ----
                        Recent      Total  PerMax
Client Errors                0          0       0
Client IPC Errors            0          0       0
Open Streams Failed          0          0       0
Drop Streams Failed          0          0       0
Exp Flow Events              0          0       0
Exp Flow Not Found           0          0       0
Transaction Timeouts         0          0       0
Server Errors                0          0       0
Server IPC Errors            0          0       0
Flow Add Failed            180        180     180
Flow Delete Failed           0          0       0
Flow Update Failed           0          0       0
Flow Latch Failed            0          0       0
Pending Flow Expired         0          0       0
ARP Wait Errors              0          0       0
Exp CAM Not Found            0          0       0
Drop Unknown Exp Flow        0          0       0
Drop/Exp Flow Missing        0          0       0
Exp Notify Failed            0          0       0
Unacknowledged Notify        0          0       0
Invalid Realm                0          0       0
No Ports Available           0          0       0
Insufficient Bandwidth       0          0       0
```

```
Stale Ports Reclaimed        0          0       0
Stale Flows Replaced         0          0       0
Telephone Events Gen         0          0       0
Pipe Alloc Errors            0          0       0
Pipe Write Errors            0          0       0
Not Found In Flows           0          0       0
XCode Internal Errors        0          0       0
XCode Alloc Errors           0          0       0
XCode Update Errors          0          0       0
XCode Delete Errors          0          0       0
XCode Over Cap Errors      180        180     180
XCode Over License Cap       0          0       0
SRTP Capacity Exceeded       0          0       0
```

show xcode api-stats

The **show xcode api-stats** command shows the client and server side message counts for the XClient and XServer software components. The main messages are allocate, update, and free of the transcoding resource. The command uses a 100 second window to show recent counts within the sliding window as well as the total and per max (maximum in a sliding window interval). This command is useful for comparing the client and server side counts and seeing where errors may have occurred with the transcoding resources.

```
ORACLE#show xcode api-stats
                     --------- Client --------   --------- Server --------
Message/Event        Recent      Total  PerMax   Recent      Total  PerMax
                     ------  ---------  ------   ------  ---------  ------
Allocs                    0       5197    4897        0       6355    6055
Updates                   0       1776    1676        0        888     788
Frees                     0       6355    6015        0       6355    6015
Error-Allocs              0          0       0        0         45      45
Error-Updates             0          0       0        0        888     888
Error-Frees               0          0       0        0          0       0
Total                     0      13328   12588        0      14531   13791
```

show xcode dbginfo

The debug information command shows the packet API statistics for the host to DSP path. There is one session/connection opened with each DSP. The command displays the total packet counts as well as the round trip time statistics for the packets. The recent field shows the count since the last time the command was executed

```
ORACLE#show xcode dbginfo
Startup Time    : 2006-09-08 01:11:50.522
Last Clear Time : 2006-09-08 01:11:50.522
Last Read Time  : 2006-09-08 17:14:52.351
Current Time    : 2006-09-08 17:14:52.351
Up Time         : 0 Days, 16 Hours 3 Minutes 2 Seconds
                          -- Life Time --    -- Recent --
PktApiStats:
    OpenConnectionCnt    =       2
    OpenSessionCnt       =       2
    TotalPktSentCnt      =   21051              21051
    TotalPktRecvCnt      =   21041              21041
    TotalPktRecvEventCnt =       0                  0
    TotalPktRecvDataCnt  =       0                  0
    TotalPktRejectCnt    =       5                  5
    TotalPktTimeoutCnt   =       0                  0
    TotalPktInvalidCnt   =       0                  0
```

```
TotalPktDropCnt      =       0                   0
TotalPktDropEventCnt =       0                   0
TotalPktDropDataCnt  =       0                   0
TotalPktLateRspCnt   =       0                   0
LowestRoundTripMs             =       1
HighestRoundTripMs            =    2010
LowestExtractTimeMs           =       1
HighestExtractTimeMs          =   13320
HighestTransportRxTimeMs      =       0
ulHighestTransportNoRxTimeMs  =       0
```

Active and Period Statistics for EVRC and other Codecs

Codec use per-realm statistics include more detail by breaking out the EVRC codecs into their specific variants. That is, the system keeps track of EVRC0, EVRC1, EVRCB, EVRCB0, and EVRCB1 codec use per-realm on an explicit basis. These 5 counts are also available for SNMP query and are added to the `ap-codec.mib` file. Additionally, The Oracle USM now maintains counts for all codecs that appear in the **show sipd codecs <realm>** command for Active, Recent High, and Lifetime High periods.

show sipd codecs

The **show sipd codecs <realm ID>** command displays media-processing statistics per SIP traffic. This command displays statistics per realm and requires a realm argument.

Session Based Statistics

The first 3 statistics listed by the **show sipd codecs** are session based. These statistics are titled Transcoded, Transrated, and Transparent.

- transcoded—counts of sessions that use the Transcoding NIU's TCUs to transcode between two or more codes.
- transrated—counts of sessions that use the Transcoding NIU's TCUs to change the packetization interval among dialogs in the session.
- transparent—counts of sessions that require no TCU hardware intervention (all end-to-end media uses the same codec)

A value of "none" which is not counted in the statistics is set when there is no media at all or media is not yet negotiated. Sessions within the same realm are counted only once.

These are meter type counters, and thus have an "active" count as well as total lifetime values. The media-processing state of the session only can increase in precedence (highest=transcoded, transrated, transparent, lowest=none). Thus, if a session begins as transcoded, and then is re-negotiated to transparent later by a re-INVITE, it is still considered transcoded. However, if a session begins as transparent, it can go to transcoded by a re-INVITE. In such a case, the total counts for both transparent and transcoded would be incremented. If there are several media lines, the highest precedence is used for the session.

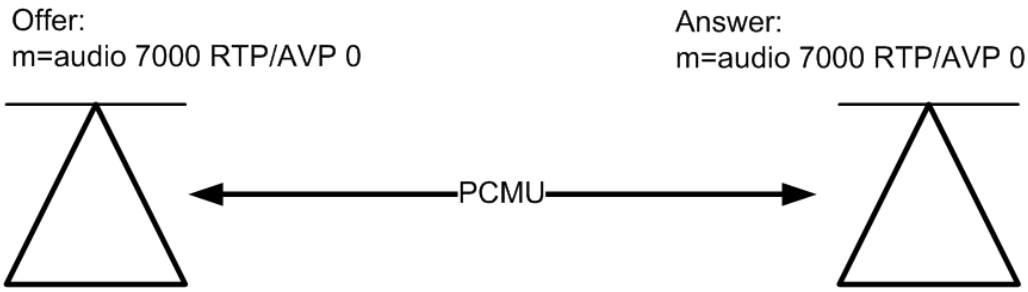
Flow Based Statistics

The remaining lines of the **show sipd codecs** command track the number of codecs in established sessions. The 'Other' type refers to unknown codecs. For all codecs listed by the **show sipd codecs** command, the Active, High- and Total- Recent Counts, and Total- PerMax-

High- Lifetime counts are displayed. These counts represent each SDP m= line emanating in the queried realm. Refer to the following examples:

Single audio stream example

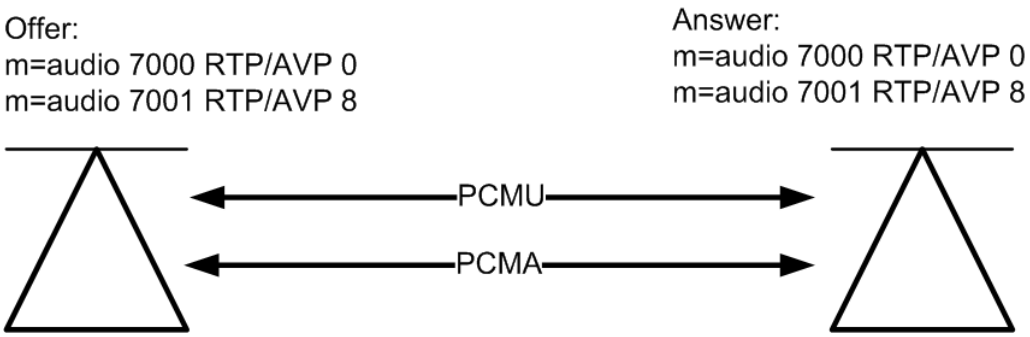
The following diagram shows an intra-realm session with one audio stream using the PCMU codec. Once the session is established, the PCMU count in the **show sidp codecs** output is 2.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count will be reflected in each respective query because only one m= line emanates from each realm.

Multiple audio stream example

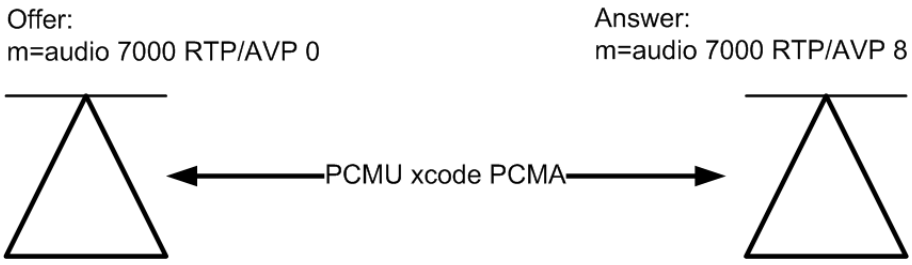
The following diagram shows an intra-realm session with two audio streams. Each stream uses a different codec. Once the session is established, the PCMU count in the **show sidp codecs** output is 2, and the PCMA count is 2.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count and a single PCMA count will be reflected in each respective query because two m= lines emanate from each realm.

Transcoded audio stream example

The following diagram shows an intra-realm transcoding scenario where the originator and terminator are using different audio codecs. The Oracle USM transcodes the media, which is invisible to the endpoints. Once the session is established, the PCMU count in the **show sidp codecs** output is 1, and the PCMA count is 1.



If the session originator and terminator in the previous diagram exist in two different realms, you must execute the **show sidp codecs** command twice, once for each realm. A single PCMU count appears in one query and a single PCMA count appears in the other query because only one m= line emanate from each realm.

An example **show sipd codecs <realm ID>** command follows:

```
ORACLE# show sipd codecs net172
09:20:04-30 Realm net172
Codec Statistics
                        ---- Recent ---- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Transcoded             0       0       0          0       0       0
Transrated             0       0       0          0       0       0
Transparent            0       0       0          0       0       0
PCMU Count             0       0       0          0       0       0
PCMA Count             0       0       0          0       0       0
G722 Count             0       0       0          0       0       0
G723 Count             0       0       0          0       0       0
G726-16 Count          0       0       0          0       0       0
G726-24 Count          0       0       0          0       0       0
G726-32 Count          0       0       0          0       0       0
G726-40 Count          0       0       0          0       0       0
G728 Count             0       0       0          0       0       0
G729 Count             0       0       0          0       0       0
GSM Count              0       0       0          0       0       0
iLBC Count             0       0       0          0       0       0
H261 Count             0       0       0          0       0       0
H263 Count             0       0       0          0       0       0
T38 Count              0       0       0          0       0       0
AMR Count              0       0       0          0       0       0
AMR-WB Count           0       0       0          0       0       0
EVRC Count             0       0       0          0       0       0
EVRC0 Count            0       0       0          0       0       0
EVRC1 Count            0       0       0          0       0       0
EVRCB Count            0       0       0          0       0       0
EVRCB0 Count           0       0       0          0       0       0
EVRCB1 Count           0       0       0          0       0       0
Other Count            0       0       0          0       0       0
```

show xcode load

The **show xcode load** command shows the current transcoding module (TCM) load both in number of sessions and percent loading. The load percentage depends on the precise mix of codecs, ptimes, and features enabled on the active sessions. The maximum lifetime load is also displayed. Uninstalled TCMs are marked with dashes.

```
ORACLE#show xcode load
17:19:11
Total Sessions:             0
```

```
Licensed AMR Sessions:       0
Licensed AMR-WB Sessions:    0
Licensed EVRC Sessions:      0
Licensed EVRCB Sessions:     0
                   ----- Load -----
  TCU  TCM  #Sess  Current  Maximum
  ===  ===  =====  =======  =======
   0    00     -       -        -
   0    01     -       -        -
   0    02     -       -        -
   0    03     0     0.00%   99.81%
   0    04     0     0.00%   99.81%
   0    05     -       -        -
   0    06     -       -        -
   0    07     -       -        -
```

The TCU column is populated with a 0 for a TCM in the middle slot and a 1 for a TCM in the top slot.

show xcode session-all

The **show xcode session-all** command displays all of the currently active sessions by their unique session id.

```
ORACLE#show xcode session-all
15:22:51
Requesting xclient sessions table
        Total Active Sessions: 200
        Displaying sessions 1 to 100:
        Session Id: 0x10007
        Session Id: 0x10008
        Session Id: 0x10009
        Session Id: 0x1000a
        Session Id: 0x1000b
```

> **Note:**
>
> When there are more than 100 active sessions , the command now displays only active sessions 1 to 100 as opposed to all the active session:

show xcode session-byid

The session-byid command gives more detailed information about the session. The session-byid command displays the configuration of each channel as well as a number of packet statistics for each channel. This same information can be looked up by IP address and port by using the session-byip command. If only the configuration portion is required, use the session-config command with the session id as the argument. This command is entered as:

```
show xcode     session-byid <session_id>
```

For example:

```
ORACLE#show xcode session-byid 0xf006e
################## SESSION  0xf006e ##################
Channel 0:
  DSP device          = 14
  Source MAC          = 00:08:25:a0:9a:f3
```

```
  Destination MAC     = 00:0e:0c:b7:32:e2
  VLAN ID             = 0
  Egress Interface    = 0
  Src IP:Port         = 172.16.0.235:24448
  Dst IP:Port         = 172.16.0.87:16000
  Src RTCP IP:Port    = 172.16.0.235:24449
  Dst RTCP IP:Port    = 172.16.0.87:16001
  Codec               = G711_ULAW_PCM
  Payload Type        = 0
  Pkt Interval        = 20 msec
  2833 Payload Type   = DISABLED
  Xtone Mode          = XTONE_XTHRU
  Status              = DISABLED
DSP Counters:
  RxInPktCnt                 474
  RxInByteCnt                75840
  RxOutPktCnt                749
  RxInSidPktCnt              0
  RxNoPktCnt                 275
  RxBadPktTypeCnt            0
  RxBadRtpPayloadTypeCnt     0
  RxBadPktHdrFormatCnt       0
  RxBadPktLengthCnt          0
  RxMisorderedPktCnt         0
  RxBadPktChecksumCnt        0
  RxUnderrunSlipCnt          0
  RxOverrunSlipCnt           0
  RxLastVocoderType          0
  RxVocoderChangeCnt         0
  RxMaxDetectedPdv           168
  RxDecdrRate                15
  RxJitter:
    CurrentDelay             160
    EstimatedDelay           0
    ClkDriftingDelta         0
    ClkDriftingCorrectionCnt 0
    InitializationCnt        1
  RxCircularBufferWriteErrCnt 0
  RxApiEventCnt              0
  TxCurrentVocoderType       0
  TxInPktCnt                 749
  TxOutPktCnt                750
  TxOutByteCnt               120000
  TxInBadPktPayloadCnt       0
  TxTimestampGapCnt          0
  TxTdmWriteErrCnt           0
  RxToneDetectedCnt          0
  RxToneRelayEventPktCnt     0
  RxToneRelayUnsupportedCnt  0
  TxToneRelayEventPktCnt     0
  TxApiEventCnt              0
  TxNoRtpEntryPktDropCnt     0
  ConnectionWaitAckFlag      1
  RxMipsProtectionDropCnt    0
  TxMipsProtectionDropCnt    0
Channel 1:
  DSP device          = 14
  Source MAC          = 00:08:25:a0:9a:f4
  Destination MAC     = 00:1b:21:7a:29:b1
  VLAN ID             = 0
  Egress Interface    = 2
```

```
  Src IP:Port         = 192.168.0.235:24448
  Dst IP:Port         = 192.168.0.87:32000
  Src RTCP IP:Port    = 192.168.0.235:24449
  Dst RTCP IP:Port    = 192.168.0.87:32001
  Codec               = G729_A
  Payload Type        = 18
  Pkt Interval        = 20 msec
  2833 Payload Type   = DISABLED
  Xtone Mode          = XTONE_XTHRU
  Status              = DISABLED
DSP Counters:
  RxInPktCnt                  748
  RxInByteCnt                 14960
  RxOutPktCnt                 751
  RxInSidPktCnt               0
  RxNoPktCnt                  3
  RxBadPktTypeCnt             0
  RxBadRtpPayloadTypeCnt      0
  RxBadPktHdrFormatCnt        0
  RxBadPktLengthCnt           0
  RxMisorderedPktCnt          0
  RxBadPktChecksumCnt         0
  RxUnderrunSlipCnt           0
  RxOverrunSlipCnt            0
  RxLastVocoderType           6
  RxVocoderChangeCnt          0
  RxMaxDetectedPdv            171
  RxDecdrRate                 15
  RxJitter:
    CurrentDelay              160
    EstimatedDelay            0
    ClkDriftingDelta          0
    ClkDriftingCorrectionCnt  0
    InitializationCnt         1
  RxCircularBufferWriteErrCnt 0
  RxApiEventCnt               0
  TxCurrentVocoderType        6
  TxInPktCnt                  748
  TxOutPktCnt                 748
  TxOutByteCnt                14960
  TxInBadPktPayloadCnt        0
  TxTimestampGapCnt           0
  TxTdmWriteErrCnt            0
  RxToneDetectedCnt           0
  RxToneRelayEventPktCnt      0
  RxToneRelayUnsupportedCnt   0
  TxToneRelayEventPktCnt      0
  TxApiEventCnt               0
  TxNoRtpEntryPktDropCnt      0
  ConnectionWaitAckFlag       0
  RxMipsProtectionDropCnt     0
  TxMipsProtectionDropCnt     0
```

show xcode session-byattr

The show xcode session-byattr command lists all sessions matching the specified attribute name. The only supported attribute is "fax", which will display session information only for FAX-transcoded sessions; all other attributes will return an error. For example:

```
ORACLE#show xcode session-byattr fax
17:52:17
1.   [Chan A_SRC] Ip Address: 192.168.16.1 Port: 5220
     [Chan B_SRC] Ip Address: 172.16.0.40 Port: 10230
     Session Id:0x002021d0"
2.   [Chan A_SRC] Ip Address: 192.168.16.1 Port: 3010
     [Chan B_SRC] Ip Address: 172.16.0.40 Port: 10226
     Session Id:0x00301042"
Oct 26 20:53:22.968 Total Matches:2 Total Active Sessions:2
-------------
```

show xcode session-byipp

The show xcode session-byipp command requires an IP address and port. It lists detailed information about the sessions identified by the specified IP address and port number. nformation will be provided for all transcoded call legs matching the IP address, including in both the ingress and egress directions. The show xcode session-byipp command is entered as:

```
show xcode session-byipp <ip_addr> <port_num>
```

This command displays the same information as the **session-byid** command. If a wildcard * is provided for the port number, the command will display sessions with the matching IP address only, regardless of port number.

show xcode xlist

The show xcode xlist command displays the TCU (0 = middle, 1 = top), TCM number, number of DSPs on each module, the number of active sessions, and the load percentage. It also displays the state such as Active or Boot Failure. Uninstalled TCMs are indicated by a dash.

```
ORACLE#show xcode xlist
18:22:32
  TCU  TCM  DSPs  #Sess  Load   State
  ===  ===  ====  =====  ====  =========
   0    00    2      -     -      -
   0    01    2      -     -      -
   0    02    2      -     -      -
   0    03    2      1     0%   2 Active
   0    04    2      1     0%   2 Active
   0    05    2      -     -      -
   0    06    2      -     -      -
   0    07    2      -     -      -
   [...]
   1    00    2      1     0%   2 Active
   1    01    2      0     0%   2 Active
   1    02    2      0     0%   2 Active
   1    03    2      0     0%   2 Active
   1    04    2      0     0%   2 Active
   1    05    2      0     0%   2 Active
   1    06    2      0     0%   2 Active
   1    07    2      0     0%   2 Active
   1    08    2      0     0%   2 Active
```

Logs

A log file named `log.xserv` can be used for debugging the transcoding feature. This log records the API between the host software and the DSPs and any errors that are encountered.

Alarms

The transcoding feature employs several hardware and software alarms to alert the user when the system is not functioning properly or overload conditions are reached.

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
No DSPs Present with Transcoding Feature Card (DSP_NONE_PRESENT)	Minor/0	A transcoding feature card is installed but no DSP modules are discovered.	apSysMgmtHardwareErrorTrap
DSP Boot Failure (DSP_BOOT_FAILURE)	Critical/0	A DSP device fails to boot properly at system initialization. This alarm is not health affecting for a single DSP boot failure. DSPs that fail to boot will remain uninitialized and will be avoided for transcoding.	apSysMgmtHardwareErrorTrap
DSP Communications Timeout (DSP_COMMS_TIMEOUT)	Critical/100	A DSP fails to respond after 2 seconds with 3 retry messages. This alarm is critical and is health affecting.	apSysMgmtHardwareErrorTrap
DSP Alerts (DSP_CORE_HALT)	Critical/100	A problem with the health of the DSP such as a halted DSP core. The software will attempt to reset the DSP and gather diagnostic information about the crash. This information will be saved in the /code directory to be retrieved by the user.	apSysMgmtHardwareErrorTrap
DSP Temperature(DSP_TEMPERATURE_HIGH)	Clear 85°C Warning 86°C / 5 Minor 90°C / 25 Major 95°C/ 50 Critical 100°C/ 100	A DSP device exceeds the temperature threshold. If the temperature exceeds 90°C, a minor alarm will be set. If it exceeds 95°C, a major alarm will be set. If it exceeds 100°C, a critical alarm will be set. The alarm is cleared if the temperature falls below 85°C. The alarm is health affecting.	apSysMgmtHardwareErrorTrap

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
Transcoding Capacity Threshold Alarm (XCODE_UTIL_OVER_THRESHOLD) / 131329	Clear 80% Warning 95%	A warning alarm will be raised when the transcoding capacity exceeds a high threshold of 95%. The alarm will be cleared after the capacity falls below a low threshold of 80%. This alarm warns the user that transcoding resources are nearly depleted. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed AMR Transcoding Capacity Threshold Alarm/ 131330	Clear 80% Warning 95%	A warning alarm is triggered if the AMR transcoding capacity exceeds a high threshold of 95% of licensed session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed AMR-WB Transcoding Capacity Threshold Alarm/ 131331	Clear 80% Warning 95%	A warning alarm is triggered if the AMR-WB transcoding capacity exceeds a high threshold of 95% of licensed session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap
Licensed EVRC Transcoding Capacity Threshold Alarm/ 131332	Clear 80% Warning 95%	A warning alarm is triggered if the EVRC transcoding capacity exceeds a high threshold of 95% of licensed session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap

Name/ID	Severity/ Health Degredation	Cause(s)	Traps Generated
Licensed EVRCB Transcoding Capacity Threshold Alarm/ 131333	Clear 80% Warning 95%	A warning alarm is triggered if the EVRCB transcoding capacity exceeds a high threshold of 95% of licensed session in use. The alarm clears after the capacity falls below a low threshold of 80%. This alarm is not health affecting.	apSysMgmtGroupTrap

Transcoding Capacity Traps

The Oracle USM sends the apSysMgmtGroupTrap as transcoding capacity nears its limit. This trap is sent and cleared for three conditions:

- Total DSP usage exceeds 95%
- Total AMR sessions exceed 95% of licensed capacity
- Total AMR-WB sessions exceed 95% of licensed capacity
- Total EVRC sessions exceed 95% of licensed capacity
- Total EVRCB sessions exceed 95% of licensed capacity

The apSysMgmtGroupTrap contains the condition observed (apSysMgmtTrapType) and the corresponding value reached (apSysMgmtTrapValue).

```
apSysMgmtGroupTrap         NOTIFICATION-TYPE
    OBJECTS        { apSysMgmtTrapType, apSysMgmtTrapValue }
    STATUS           current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          exceeds a certain threshold. "
    ::= { apSystemManagementNotifications 1 }
```

When the resource usage retreats below a defined threshold, the Oracle USM sends an apSysMgmtGroupClearTrap.

```
apSysMgmtGroupClearTrap         NOTIFICATION-TYPE
    OBJECTS        { apSysMgmtTrapType }
    STATUS           current
    DESCRIPTION
          " The trap will generated if value of the monitoring object
          returns to within a certain threshold.  This signifies that
          an alarm caused by that monitoring object has been cleared. "
    ::= { apSystemManagementNotifications 2 }
```

The following table summarizes trigger and clear conditions for transcoding capacity alerts as sent in the the apSysMgmtGroupTrap:

Monitored Transcoding Resource	SNMP Object & OID in apSysMgmtTrapType	Trap Sent	Clear Trap Sent
Total DSP Usage	apSysXCodeCapacity 1.3.6.1.4.1.9148.3.2.1.1.34	95%	80%
AMR License Capacity Usage	apSysXCodeAMRCapacity 1.3.6.1.4.1.9148.3.2.1.1.35	95%	80%
AMR-WB License Capacity Usage	apSysXCodeAMRWBCapacity 1.3.6.1.4.1.9148.3.2.1.1.36	95%	80%
EVRC License Capacity Usage	apSysXCodeEVRCCapacity 1.3.6.1.4.1.9148.3.2.1.1.39	95%	80%
EVRCB License Capacity Usage	apSysXCodeEVRCBCapacity 1.3.6.1.4.1.9148.3.2.1.1.40	95%	80%

The following SNMP Objects are inserted into the apSysMgmtTrapType when sending and clearing a transcoding capacity trap. You mayt query them individually with an SNMP GET.

- apSysXCodeCapacity (1.3.6.1.4.1.9148.3.2.1.1.34)
- apSysXCodeAMRCapacity (1.3.6.1.4.1.9148.3.2.1.1.35)
- apSysXCodeAMRWBCapacity (1.3.6.1.4.1.9148.3.2.1.1.36)
- apSysXCodeEVRCCapacity (1.3.6.1.4.1.9148.3.2.1.1.39)
- apSysXCodeEVRCBCapacity(1.3.6.1.4.1.9148.3.2.1.1.40)

SNMP

The following sections have been removed from this document. Their content has been reformatted and appears in the MIB Reference guide in the *Codec and Transcoding MIB (ap-codec.mib)* section:

- SNMP Retrieval of Transcoding Statistics
- Acme Packet Codec and Transcoding MIB (ap-codec.mib)
- Acme Packet System Management MIB (ap-smgmt.mib)

Generating RTCP

The Oracle USM is capable of creating and sending RTCP reports using Transcoding resources. This produces RFC 3550 compliant RTCP report information on media traffic for active sessions. The system calculates these statistics using measurements on traffic between a target end station and itself. With respect to a given media session, the system does not produce end-to-end reports. In addition, the system can drop RTCP reports generated upstream so target stations don't receive RTCP reports that are redundant to its own.

RTCP reporting generated at the Oracle USM provides the following benefits:

- Provide RTCP reporting to core applications that may otherwise not be receiving this data.
- Provide RTCP reporting to access elements, including mobile client applications, that may otherwise not be receiving this data.

- Generate valid RTCP data for transcoded calls.
- Generate RTCP data from the Oracle USM's perspective.
- Applicable RTCP reports continue to be sent when a call is muted or placed on-hold.

The user enables and specifies the type of calls on which the system generates RTCP reports via configuration. Applicable configuration includes creating **rtcp-policy** objects and applying them to realms. These objects specify whether to issue reports for transcoded or all calls traversing the realm. There is also a configuration to disable the policy.

How it Works

The Oracle USM generates RTCP by sending RTCP sender report information to the media termination point within 5 seconds after an RTP session starts, per RFC 3550. All RTCP messaging includes Sender Reports and, if the end station is receiving media, statistics corresponding to the received media. This messaging persists, within 5-second windows for the duration of the session. The system sends a goodbye message no more than 20 ms after the call is complete. The network administrator should note this timing and ensure that network NAT configuration does not block these final messages. Depending on call type and configuration, the system listens for and drops RTCP reports generated upstream. RTCP reports generated prior to transcoding provide unreliable information.

All reports include the source description with a CNAME string. The CNAME string is the IP address and port number used for the corresponding media stream's RTCP in the format:

```
<UDP Local Port Num for RTCP>@<IP address of the applicable steering pool>
```

The system sources this data from the applicable steering pool configuration.

Functional Matrix - Configuration vs. Call Type

RTCP generation and any corresponding RTCP blocking is a function of the type of call (transcoded or non-transcoded), and configuration. Applicable **rtcp-policy** configuration includes specifying whether the system should generate RTCP for all calls, or for transcoded calls only.

The behavior is as follows:

- For Non-Transcoded Calls—Based on the RTCP-generation configuration:
  - **xcoded-calls-only**—No RTCP generated; existing RTCP passes
  - **all-calls**—RTCP generated; existing RTCP blocked
- For Transcoded Calls—RTCP generated; existing RTCP blocked for both configurations
- The RTCP generation setting includes a disabled setting (**none**) that is set by default. This specifies that the policy never generate RTCP. Realms configured with these disabled policy settings cause the system to pass existing RTCP for non-transcoded calls and block existing RTCP for transcoded calls.

When RTCP generation is enabled for non-transcoded calls using the **all-calls** setting, these calls must negotiate a codec that the Oracle USM can transcode. This ensures that the call utilizes the system's transcoding resources, which can then generate the RTCP.

> **Note:**
>
> A realm's **rtcp-policy** takes precedence over its **block-rtcp** setting. Specifically, if **block-rtcp** is disabled and the **rtcp-policy** is set to block, the system blocks existing RTCP.

RTCP Generation Platform Support

RTCP generation is supported on Acme Packet 4600, Acme Packet 4500 and Acme Packet 6300 platforms when they have a transcoding module populated with DSPs.

When generating RTCP for a non-transcoded call using DSPs, the system processes the call as if it is a transcoded call, but uses a "null" transcoding methodology that sets the input codec equal to the output codec.

There are important alarms that help the user verify the status of this hardware. Refer to the Transcoding Troubleshooting and Maintenance section in this guide for more information about transcoding hardware status. Refer to the *Maintenance and Troubleshooting Guide* for information about interface hardware status.

There is no specific license required to generate RTCP reports.

Users can determine that they have a hardware configuration that supports RTCP generation as follows using the **show xcode xlist** command to verify the presence of transcoding hardware with DSP modules.

Configuring RTCP Generation

The procedure below provides the steps needed to configure RTCP generation on the Oracle USM.

Before proceeding, make sure you know whether your platform supports transcoding. As described in the section on RTCP generation, this can affect your configuration and operational results.

To have your Oracle USM generate RTCP traffic statistics reporting:

1. In Superuser mode, use the following command sequence to access the **rtcp-policy** element.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# rtcp-policy
   ```

   From this point, you can configure a new RTCP policy or select an existing policy for editing.

2. **name**—Name your policy for use when applying it to a **realm-config**.

   ```
   ORACLE(rtcp-policy)# name report-allcalls
   ```

3. **rtcp-generate**—Enter the desired setting to generate RTCP. The effect of these settings is dependent on the platform and the presence of specific hardware on that platform. See the section on Generating RTCP in the ACLI Configuration Guide for these details.

   - **none**—Disables this policy.

- **all-xcoded-calls**—The system generates RTCP report information only for the transcoded calls that pass through the realm.
- **all-calls**—The system generates RTCP report information for all calls that pass through the realm.

```
ORACLE(rtcp-policy)# rtcp-generate all-calls
```

4. Type **done** and **exit** to retain your configuration.
5. Access the realm to which you want to apply this rtcp-policy.

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)# select
```

6. Apply the rtcp-policy to your realm.

```
ORACLE(realm-config)# rtcp-policy report-allcalls
```

7. Type **done** and exit configuration mode. Save and activate your configuration.

Obtaining System Information about RTCP Generation

The Oracle USM provides information about RTCP report generation within log files.

The **log.mbcd** file, when configured to capture at the DEBUG level, includes system messages related to invoking and firing **rtcp-policy** rules.

An example of applicable log information is provided below.

```
[XC] NatFlow::check_transcoding()
[XC]  SessionSetRtcpPorts for A: src=10001 dest=20001
[XC] NatFlow::check_transcoding() SessionSetRtcpOptions(A):
[XC]  RtcpPolicy "my_rtcp_policy"; rtcp-generate set to: RTCP_GEN_MODE_ALL_CALLS
[XC]  Will generate RTCP with CNAME=192.160.1.100:10001
[XC] NatFlow::check_transcoding()
[XC]  SessionSetRtcpPorts for B: src=10001 dest=20001
[XC] NatFlow::check_transcoding() SessionSetRtcpOptions(B):
[XC]  RtcpPolicy "my_rtcp_policy"; rtcp-generate set to: RTCP_GEN_MODE_ALL_CALLS
[XC]  Will generate RTCP with CNAME=172.16.1.100:10101
```

Error and non-error system messages related to RTCP generation can be found in **log.sipd**, **log.mbcd**, and **log.xserv**.

Acme Packet 6300 NIU Hotswap Guidelines

TheOracle USM 6300 provides a 3-slot chassis. Each chassis is monitored by a dedicated Hot-Swap sensor that maintains state information for the resident Network Interface Units.

The Oracle USM 6300 is the first multi-slot SBC that runs C series software. A hotswap sensor, located on each of the 6300's three slots, provides the ability to track the state of individual network interface units (NIUs) in the 6300 chassis.

There are 8 possible associated states as shown below.

```
PRESENT/NOT PRESENT       Checks the physical presence of a NIU inserted in a
6300 slot
LATCH OPEN/LATCH CLOSED   Checks the physical latch in the back of a NIU
POWER GOOD/ POWER BAD     Checks the power conditions of an NIU
READY/NOT READY           Checks all of the three previous state
```

A fully functional NIU is always in the following state: present, latch closed, power good, and ready.

The status of each NIU can be check by issuing the command **show platform hotswap** from the ACLI.

Network Interface Unit Removal/Replacement -- Standalone Node

A user or field engineer can remove a Network Interface Unit (NIU) from the 6300 chassis. When an NIU is removed, all services associated with the NIU (for example, media transmission, signaling, transcoding, and so on) will be interrupted and the system will be out-of-service (OOS).

Use the following procedure to remove an NIU.

1. Working from the back of the chassis, carefully open the NIU's latch. The hotswap LED placed directly above the NIU will transition from green (present/latch close/ready) to blinking blue. After 10 seconds it will stop blinking and remain blue (present/latch open/not ready).
2. You can now safely remove your NIU.

Removing the NIU generates a critical alarm notifying the user that the NIU has been removed. Additionally, the Vacuum Flourescent Display (VFD) displays a hardware alarm and starts blinking.From the ACLI, issue a "reboot force" command.

To see the alarm use: "display-alarms" from the ACLI.

To replace an NIU, insert an NIU of a different or the same type as the one used before.

The alarm manager clears the critical alarm, and issues new minor alarm to alert the user that an NIU was inserted after the boot sequence. At this point the system is still OOS.

After rebooting the system, service will be restored, and the new NIU will be functional

NIU Removal/Replacement -- High Availability Deployment

You cannot remove an NIU from the chassis of a 6300 SBC functioning in the active mode. If an NIU is accidentally removed, (1) all services will be interrupted; (2) a critical alarm will be triggered to notify the user that the NIU has been removed; (3) the system will relinquish its active state; and (4) proceed to reboot itself.

Under this scenario, service will not be interrupted because the standby node will assume the active role

In a 6300 functioning in standby mode, the user or field engineer can remove an NIU from the chassis. All services associated with the NIU (for example, media transmission, signaling, transcoding, and so on) will be interrupted and the standby node will be OOS. Removal of an NIU from the standby, however, will not affect the active node which remains in service.

Use the following procedure to remove an NIU from the standby node.

1. Working from the back of the chassis, carefully open the NIU's latch. The hotswap LED placed directly above the NIU will transition from green (present/latch close/ready) to blinking blue. After 10 seconds it will stop blinking and remain blue (present/latch open/not ready)

2. You can now safely remove the NIU. To replace an NIU, insert an NIU of the same type as the one used before.

Removing the NIU generates a critical alarm notifying the user that the NIU has been removed. Additionally, the VFD displays a hardware alarm and starts blinking

To see the alarm use: "display-alarms" from the ACLI.

The alarm manager clears the critical alarm, and issues a new minor alarm to alert the user that an NIU was inserted after the boot sequence. At this point the system is still OOS.

From the ACLI, issue a "reboot force" command.

After rebooting the standby, it will revert to standby mode.

20

DTMF Transfer and Support

DTMF Interworking

Multimedia devices and applications can exchange user-input DTMF information end-to-end over IP networks. The Oracle USM provides the capabilities required to interconnect networks and devices that use different DTMF indication signaling protocols.

DTMF Indication

There are three ways to convey DTMF information for packet-based communications:

- DTMF audio tones: DTMF digit waveforms are encoded inline with voice packets. This method only works with uncompressed audio codecs like G.711. Compressed audio codecs like G.729 and G.723 are incompatible with DTMF audio. DTMF audio is also referred to as in-band tones.
- Out-of-band signaling events:
  SIP INFO messages with Content-Type: application/dtmf-relay define out-of-band signaling events for transmitting DTMF information. SIP INFO messages separate DTMF digits from the voice stream and send them in their own signaling message.
- RTP named telephony events (NTE): RFC 2833 telephone-events are a standard that describes how to transport DTMF tones in RTP packets according to section 3 of RFC 2833. Of the three RTP payload formats available, the Oracle USM supports RTP NTE.

RFC 2833 telephone-event

RFC 2833 specifies a way of encoding DTMF-indications in RTP media streams. It does not encode the audio of the tone itself, instead data represents the sent tone. RFC 2833 can be used with SIP.

RFC 2833 defines the format of NTE RTP packets used to transport DTMF digits and other telephony events between two peer endpoints. With the NTE method, the endpoints perform per-call negotiation of the DTMF transfer method. They also negotiate to determine the payload type value for the NTE RTP packets.

The NTE payload takes the place of codec data in a standard RTP packet. The payload type number field of the RTP packet header identifies the contents as 2833 NTE. The payload type number is negotiated per call. The local device sends the payload type number to use for RFC 2833 packets using SDP, which tells the other side what payload type number to use when sending the named event packets to the local device. Most devices use payload type number 101 for RFC 2833 packets, although no default is specified in the standard.

The RFC 2833 packet's RTP header also makes use of the timestamp field. Because events often last longer than the RFC 2833 packets sending interval, the timestamp of the first 2833 packet for an event represents the beginning reference time for subsequent RFC 2833 packets for that same event. For events that span multiple RTP packets, the RTP timestamp identifies the beginning of the event. As a result, several RTP packets might carry the same timestamp.

SIP INFO Messages

SIP INFO messages can send indications of DTMF audio tones between peers as part of the signaling path of the call. Upon receipt of a SIP INFO message with DTMF content, the gateway generates the specified DTMF tone on the receiving end of the call.

DTMF Transfer Processing Overview

Enabling 2 UAs to communicate different DTMF indications with each other is facilitated in two steps. First, the Oracle USM takes an active role in the SDP negotiation between two UAs, this is the capability negotiation step. You can configure your system to suggest the DTMF indication methods that each endpoint can and can not use.

The second step is translation evaluation. After the SDP negotiation is complete, based on system configuration and the media support on each call leg, the Oracle USM performs DTMF indication conversion or passive forwarding of DTMF indication messages between UAs.

Capability Negotiation

SDP capability negotiation is the first phase of enabling DTMF transfer. The completion of the SDP offer/answer exchange yields a set of supported codecs between each UA and the Oracle USM .

For DTMF transfer consideration, SDP manipulation is directed by parameters set in one of three configuration elements:

- codec policy
- signaling interface's RFC 2833 mode
- session agent's RFC 2833 mode

The Oracle USM performs SDP manipulation (addition, removal, or modification of supported codecs) toward the called party first by any applicable codec policies. If one or more of the actions which define telephone-event Modification by Codec Policy occurs, then SDP manipulation is only performed by the codec policy configuration, not by RFC 2833 mode parameters in the signaling interface or session agent.

If a codec policy attached to either the ingress or egress realm triggers SDP manipulation, then the other realm uses codec policy for any telephone-event SDP manipulation; none of the RFC 2833 Mode configurations are used for SDP manipulation.

> **Note:**
>
> If the call does not trigger the evaluation of any codec policies, all DTMF transfer processing is only subject to RFC 2833 Mode rules.

If none of the telephone-event Modification by Codec occur, then the Oracle USM performs SDP manipulation according to the RFC 2833 Mode parameters in the signaling interface or session agent.

SDP Manipulated by Codec Policy

When a call is received, any applicable codec policies are applied and evaluated. If telephone-event SDP is modified by codec policies, then SDP manipulation by RFC 2833 Mode is not performed for either side of the call.

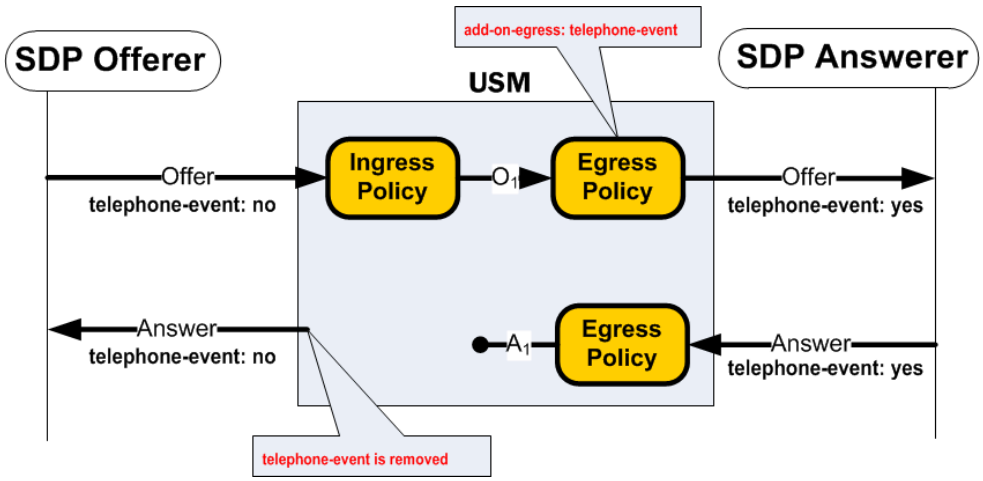
telephone-event Modification by Codec Policy

For qualifying if telephone-event modification in SDP was performed by codec policy, one of the following events had to have happened:

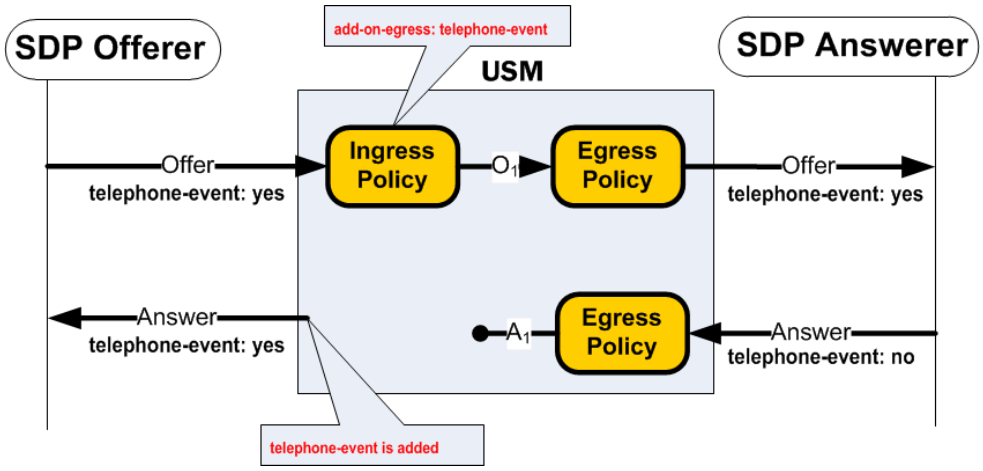
- Codec policy explicitly deleted telephone-event by configuring **allow telephone-event:no**
- Codec policy explicitly added telephone-event by configuring **add-on-egress telephone-event**
- Codec policy implicitly denied telephone-event by allowing one or more codecs but not adding telephone-event to the allow list
- Codec policy has **audio:no** configured in the allow list

The following three cases highlight how codec policies can manipulate telephone-event SDP. Once any of these cases occurs, SDP manipulation by RFC 2833 Mode parameter will not be performed.

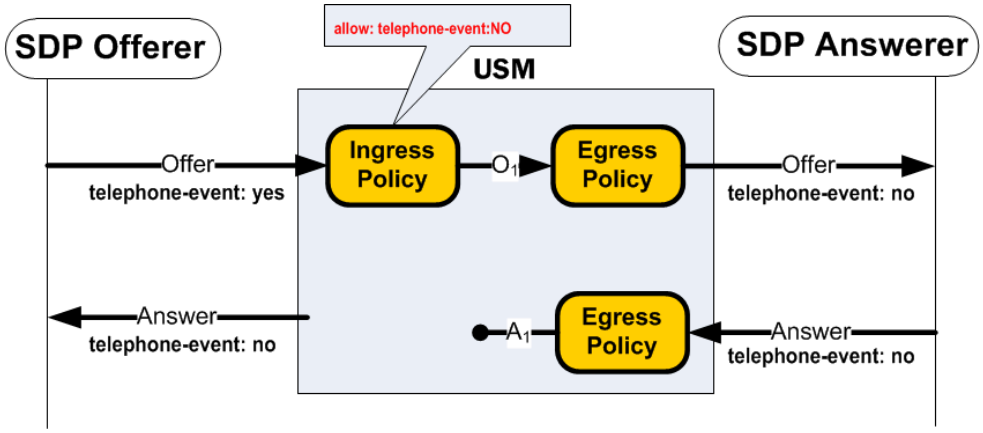
1. telephone-event added to SDP: The codec policy adds telephone-event to the SDP sent to the egress realm. The UA supports telephone-event.



2. telephone-event maintained in SDP: The codec policy maintains the offered telephone-event in the SDP sent into the egress realm. Although telephone-event was not answered in the egress realm, telephone-event is added back to the offerer-side SDP because of the add-on-egress setting in the ingress realm.



3. telephone-event removed from SDP: codec policy removes telephone-event from the initial SDP offer. telephone-event is not forwarded to the Answerer, and subsequently not returned from the answerer, or forwarded again to the offerer.



SDP Manipulated by RFC 2833 Mode

If none of the telephone-event Modification by Codec Policy events occur, as previously explained, SDP may still be modified by RFC 2833 Mode parameter if preferred or dual mode is configured.

The RFC 2833 mode parameter functions similarly to the add-on-egress parameter; it suggests telephone-event support, by adding it in SDP if not already there. This parameter must be set to preferred or dual to add telephone-event to SDP.

Transparent RFC 2833 Support

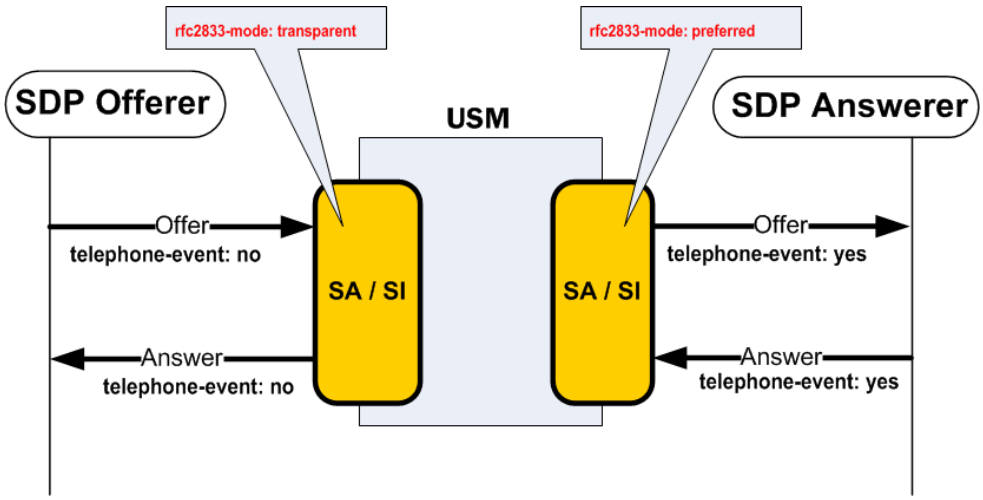
Setting a signaling interface or session agent's RFC 2833 mode to transparent disables the addition of RFC 2833 telephone-event to SDP upon egress. The Oracle USM passes the offered SDP capabilities to the next-hop signaling element.

Preferred RFC 2883 Support

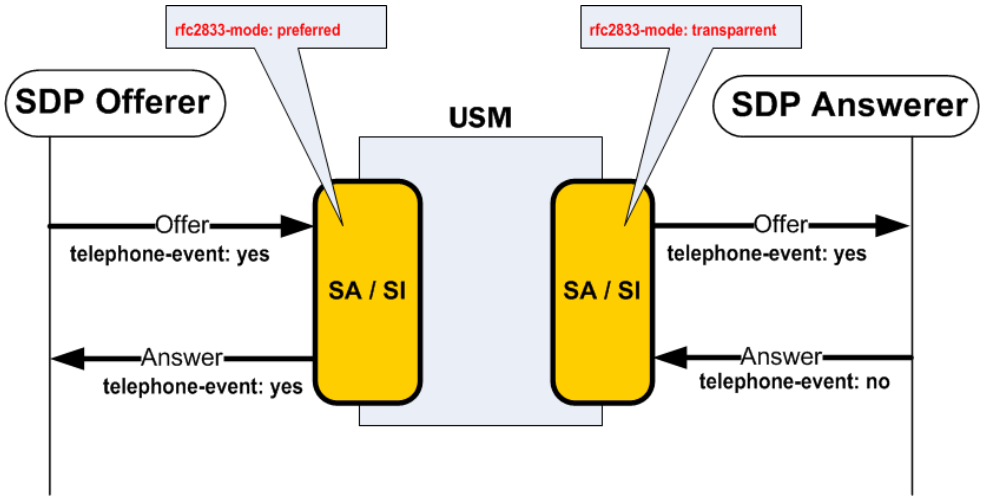
Setting a signaling interface or session agent's RFC 2833 mode to preferred indicates that the RFC 2833 telephone-event DTMF transfer method is the preferred method for sending a

DTMF indication. In the capability negotiation phase a telephone-event media type will be inserted in the outgoing SDP offer, if it was not present in the original offer.

1. In the following example RFC 2833 mode is set to preferred on the egress side of the call. Because there is no telephone-event in the SDP, and RFC 2833 mode is set to preferred, the Oracle USM adds telephone-event to the SDP offer.



2. In the following example, RFC 2833 is set to preferred mode on the SDP offer side of the call. The Oracle USM maintains the telephone-event support even though telephone-event is not supported on the SDP answerer's side of the call.



RFC 2833 Payload Type Mapping

The Oracle USM does not require that call legs use the same media type for telephone-event. If each call leg uses a different media type value, the Oracle USM facilitates payload type mapping to ensure the telephone-event media stream be reliably transported across the call.

- On the SDP offer side, when the Oracle USM returns its SDP answer, it uses the same media type that the SDP offerer offered.
- The Oracle USM forwards the originally offered telephone-event media type to the SDP answerer. If telephone-event was added by RFC 2833 mode, the Oracle USM adds

telephone-event with the media type value configured in the RFC 2833 payload parameter. If telephone-event was added by a codec policy, the Oracle USM adds telephone-event with the media type value configured in the media profile.

- If the SDP answerer returns a new value for telephone-event, the Oracle USM still supports RFC 2833 on that side of the call and uses the media type that the answerer sent.

Translation Evaluation

After SDP has been negotiated, the Oracle USM determines what types of DTMF translation takes place for the call. The Oracle USM sequentially evaluates the following rules for each call leg to determine what DTMF indication type it will forward to an endpoint.

1. RFC 2833—When the SDP offer/answer exchange resolves to both the Oracle USM and the endpoint supporting RFC 2833 on one side of the call, the Oracle USM will send DTMF indications in RFC 2833 format.
2. DTMF audio tones—Three conditions must be met for the Oracle USM to support DTMF audio tones, as transcoded from another DTMF indication form:
   - The applicable codec policy's dtmf in audio parameter is set to preferred
   - The endpoint and Oracle USM have negotiated to a DTMFable codec (G711)
   - Transcoding resources are available

   > **Note:**
   >
   > Because of rule number one, rule number two can not happen if RFC 2833 is supported in SDP—Only one media-based DTMF transfer method, RFC 2833 or DTMF audio tones may be used on a call leg.
3. If neither RFC 2833 nor DTMF Audio tones are supported on a call leg, as a result of SDP negotiation, then the Oracle USM forwards DTMF indication messages to that side in signaling message format (SIP INFO).

   In the following images that illustrate DTMF transfer scenarios, a gears icon appears when relevant. This icon indicates that the Oracle USM performs DTMF indication processing, creating DTMF audio tones or RFC 2833 telephone-event messages from another form of DTMF indication.

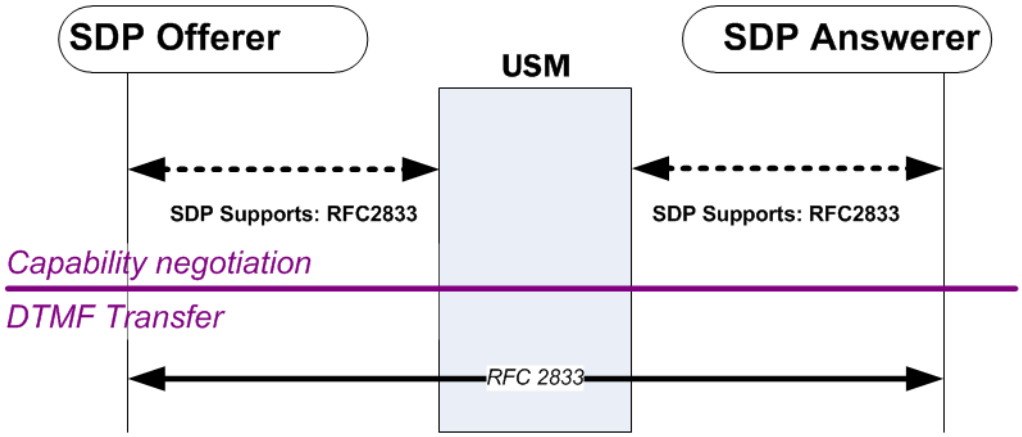
RFC 2833 Sent by Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in RFC 2833 format. The SDP answerer can receive DTMF indications in the format identified in each example.

RFC 2833 to RFC 2833

When the SDP offer and answer sides of a call both support RFC 2833, the Oracle USM forwards RFC 2833 messages between both sides of the call. No processing is used to transform these DTMF-indication massages to another format.

> **Note:**
>
> When the audio stream is transcoded, DTMF audio is completely removed from the audio stream.



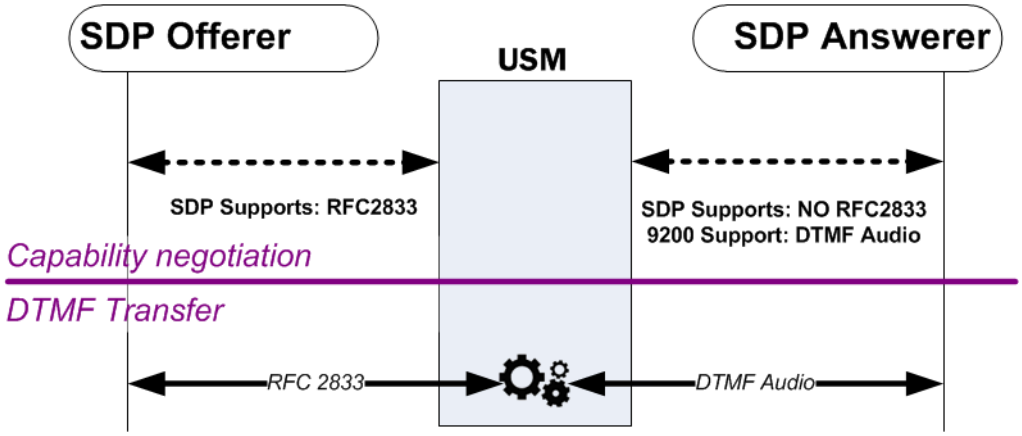
A SIP INFO message received from either the offerer or answerer is forwarded unconverted to the other side of the call.

If there is no audio transcoding enabled for this call, and the egress side is set to dual, a received SIP INFO message will not be converted to both RFC 2833 and SIP INFO messaged for sending to the other side of the call.

If DTMF audio tones are received from either the offerer or answer, they are forwarded unconverted to the other side (when the audio portion of the call is not transcoded).

RFC 2833 to DTMF Audio Tones

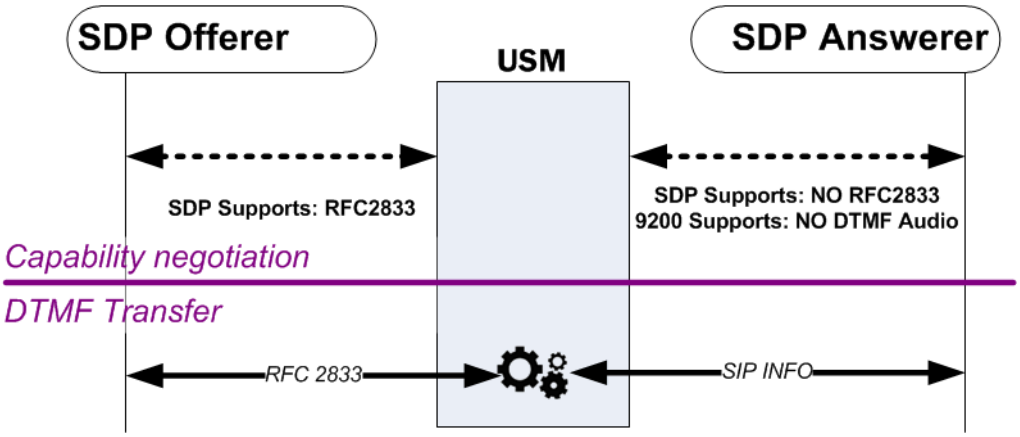
When the SDP offer side supports RFC 2833, and the SDP answer side supports the three DTMF Audio Tone conditions and does not support RFC 2833, the Oracle USM converts from RFC 2833 to DTMF audio tones for the call.



A SIP INFO message received by the Oracle USM from either the offerer or answerer is converted into the DTMF transfer method that the previous diagram shows for the egress side of the message. In this case, transcoding resources are used.

RFC 2833 to SIP INFO

When the SDP offer side supports RFC 2833 and the SDP answer side does not support the DTMF conditions nor RFC 2833, the Oracle USM converts from RFC 2833 to SIP INFO.



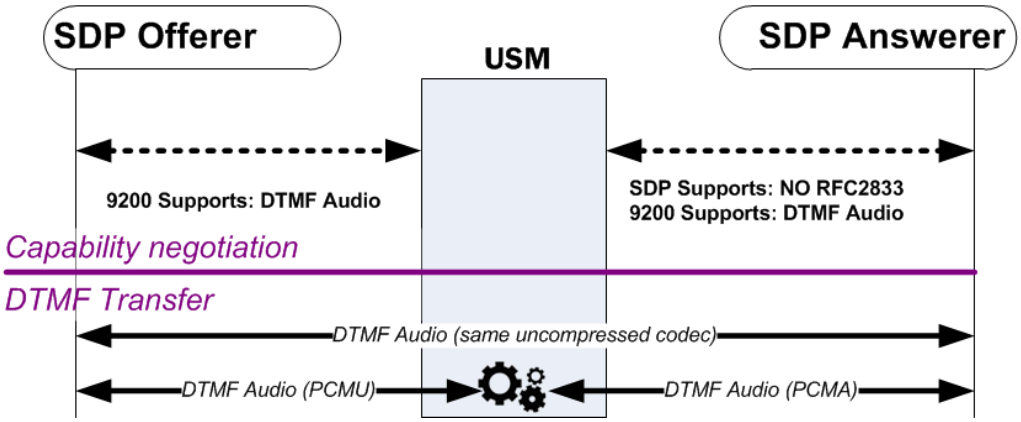
If a SIP message is received from the offerer, it is forwarded unconverted to the answerer.

DTMF Audio Tones Sent by Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in DTMF audio tones format. The SDP answerer can receive DTMF indications in the format identified in each example.

DTMF Audio to DTMF Audio

If the SDP offer and answer sides both support the same type of G711 codec, the audio stream is forwarded between the two sides without processing.

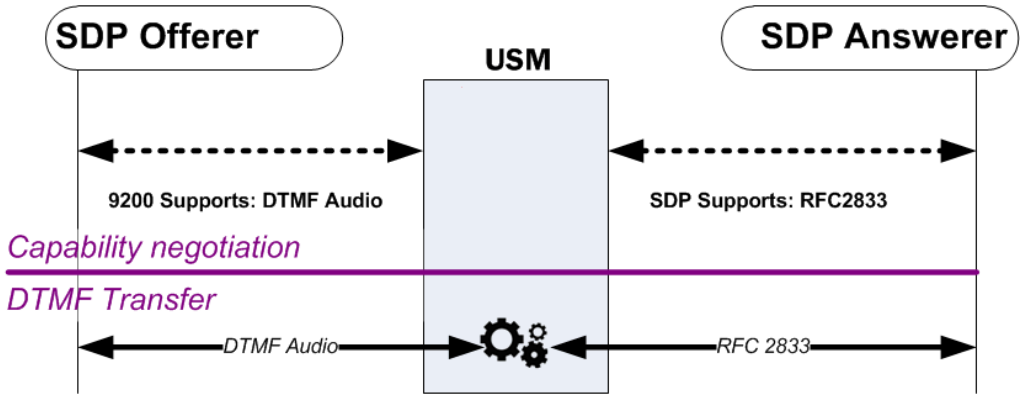


If the two sides of the call support DTMF audio tones, but use different audio codecs, and the SDP answer side supports the three DTMF Audio Tone conditions and does not support RFC 2833 then the Oracle USM will preserve DTMF audio tone indication across the call.

Transcoding resources are used only if different audio codecs are used or the Override Preferred DTMF Audio feature is enabled.

DTMF Audio to RFC 2833

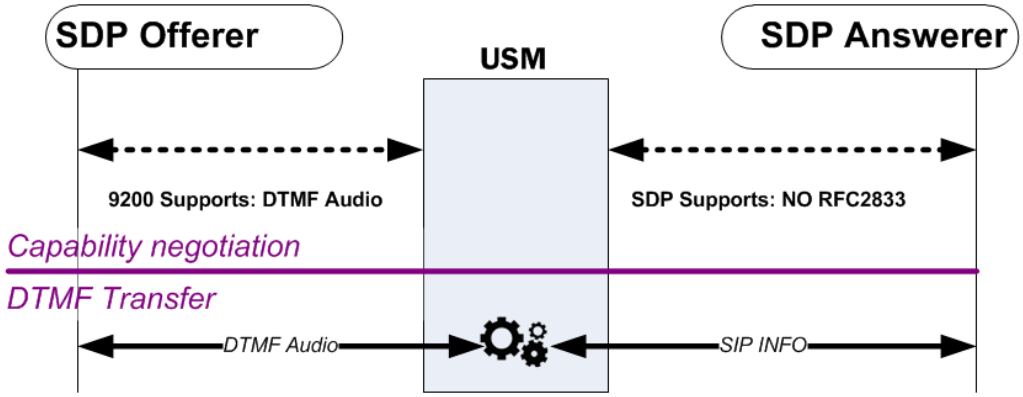
When the SDP offer side supports DTMF audio tones, and the SDP answer side supports RFC 2833, and transcoding resources are available, and does NOT support either or both of the first two DTMF Audio tone conditions, then the Oracle USM will convert incoming DTMF audio tones to outgoing RFC 2833 packets.



Transcoding resources are always required in this scenario.

DTMF Audio to SIP

When the SDP offer side supports DTMF audio tones, and the SDP answer side does not support RFC 2833, and does not support the three DTMF Audio Tone conditions, then the Oracle USM converts incoming DTMF audio tones to SIP INFOmessages.



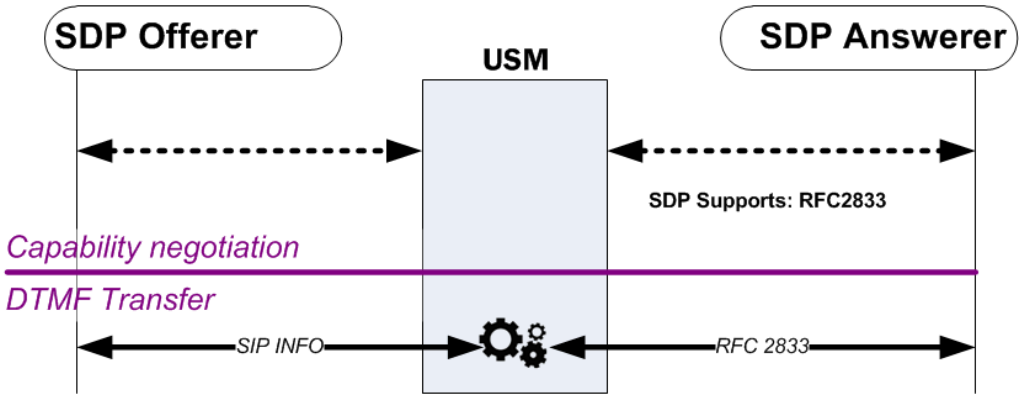
Transcoding resources are always required in this scenario.

SIP INFO Sent By Offerer

In the following three examples, the SDP offerer sends DTMF indication messages in SIP INFO message format. The SDP answerer can receive DTMF indications in the format identified in each example.

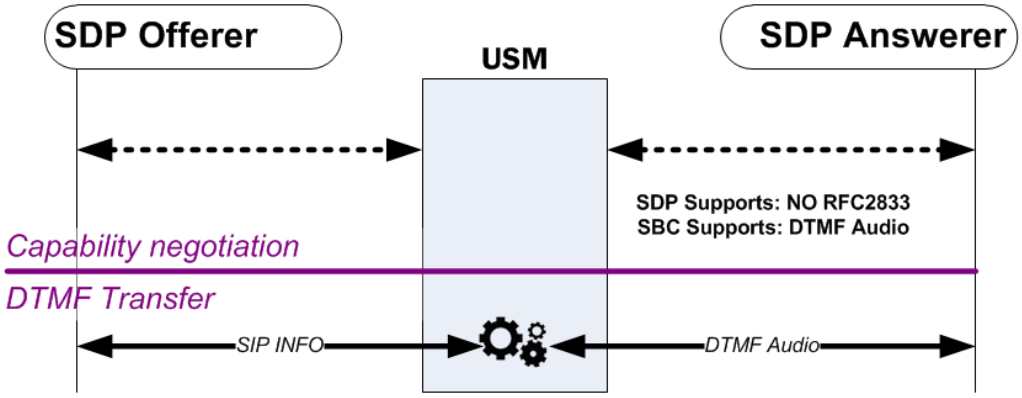
SIP INFO to RFC 2833

When the SDP offer side sends a SIP INFO message, and the SDP answer side supports RFC 2833, then the Oracle USM will convert incoming SIP INFO messages to outgoing RFC 2833 packets.



SIP INFO to DTMF Audio

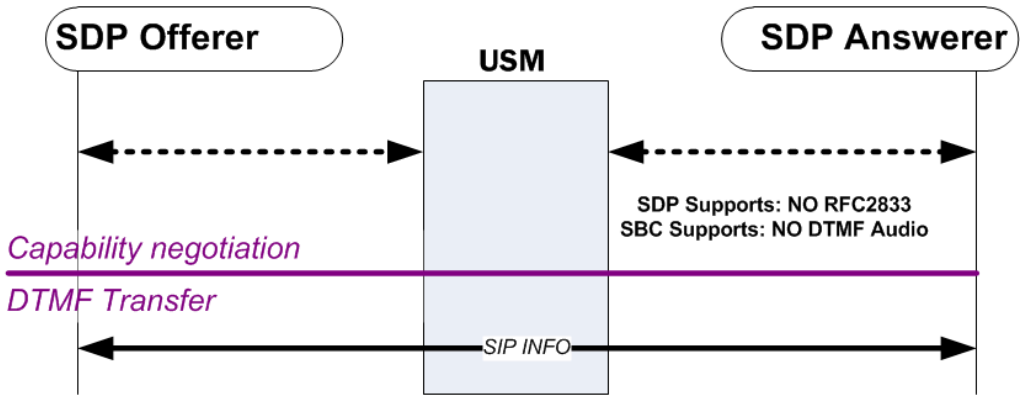
SIP INFO will only be converted to DTMF audio tones only if RFC 2833 is not supported, dtmf-in-audio is enabled, the answer side supports a G711 codec, and transcoding resources are available.



Transcoding resources are always required in this scenario.

SIP INFO to SIP INFO

When the SDP offer side sends a SIP INFO message and the SDP answer side does not support RFC 2833 and does not support the three DTMF audio tone conditions, the SIP INFO message will always be forwarded as the like SIP INFO message.



Dual Mode

Dual mode is used to send both RFC 2833 and the protocol-specific DTMF indication to a UA when possible: SIP INFO from a SIP interface.

To consider dual mode scenarios, the Oracle USM sets up the call SDP. At the conclusion of the capability negotiation, the Oracle USM is configured to support DTMF Audio tones or RFC 2833 independently for each side of the call.

When the call leg supports RFC 2833 as the means of DTMF transfer, the Oracle USM checks if the SIP interface's (or session agent's) RFC 2833-mode parameter is configured to dual. If it is, the Oracle USM sends both RFC 2833 and SIP INFO messages to the UA on this side of the call.

> **Note:**
>
> Whether RFC 2833 support was initiated between the Oracle USM and the UA by a codec policy or by the rfc2833-mode parameter, the Oracle USM looks to the rfc-2833 parameter to consider if dual mode is supported.

When the call leg supports DTMF audio tones as the means of DTMF transfer, the Oracle USM checks if the codec policy's dtmf-in-audio parameter is configured to dual. If it is, the Oracle USM sends both DTMF audio tones and SIP INFOmessages to the UA on this side of the call.

P-Dual-Info Header

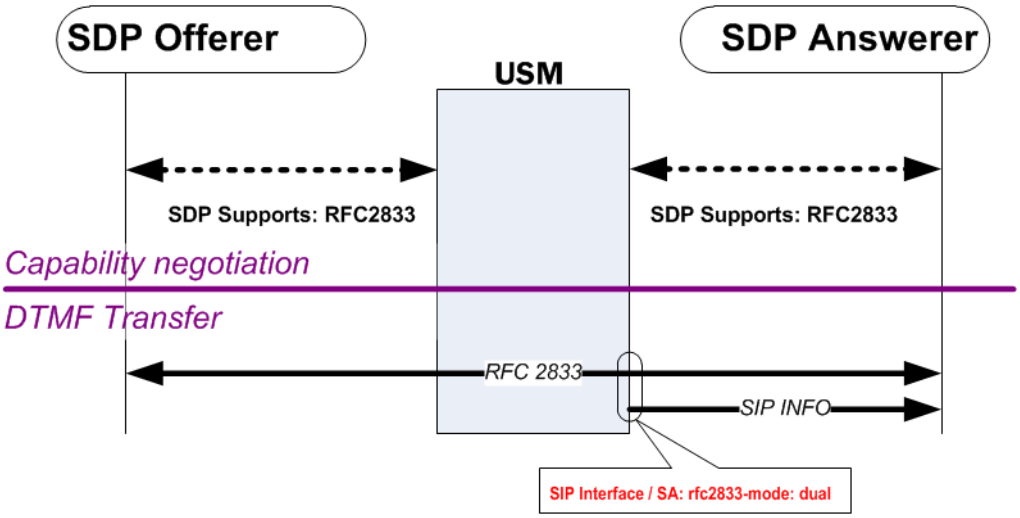
When the Oracle USM forward both media DTMF indication and signaling based DTMF indication for the same received DTMF indication, a P-Dual-Info header is added to the forwarded signaling message. You can configure the appearance of the header with the **dual-info** option. The default header appearance is:

```
P-dual-info: true
```

P-Dual-Info headers are only inserted into SIP INFO messages.

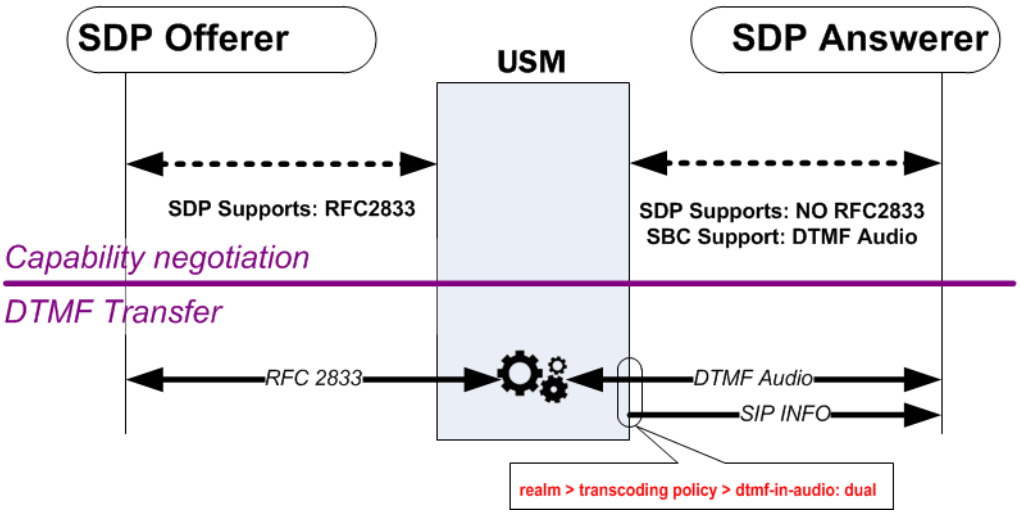
Example 1

In this example, RFC 2833 is supported on the egress side of the call. The egress SIP interface or session agent's rfc2833-mode is set to dual mode. When the Oracle USM forwards RFC 2833 to the SDP answerer, it also creates and forwards a corresponding SIP INFO message toward the target.



Example 2

In this example RFC 2833 telephone-event is not supported on the egress side of the call, but DTMF audio tones are. If the Oracle USM receives an RFC 2833 message, it is converted to DTMF audio tones. When the Oracle USM forwards DTMF audio tones to the SDP answerer, it also creates and forwards a corresponding SIP INFO message toward the target.



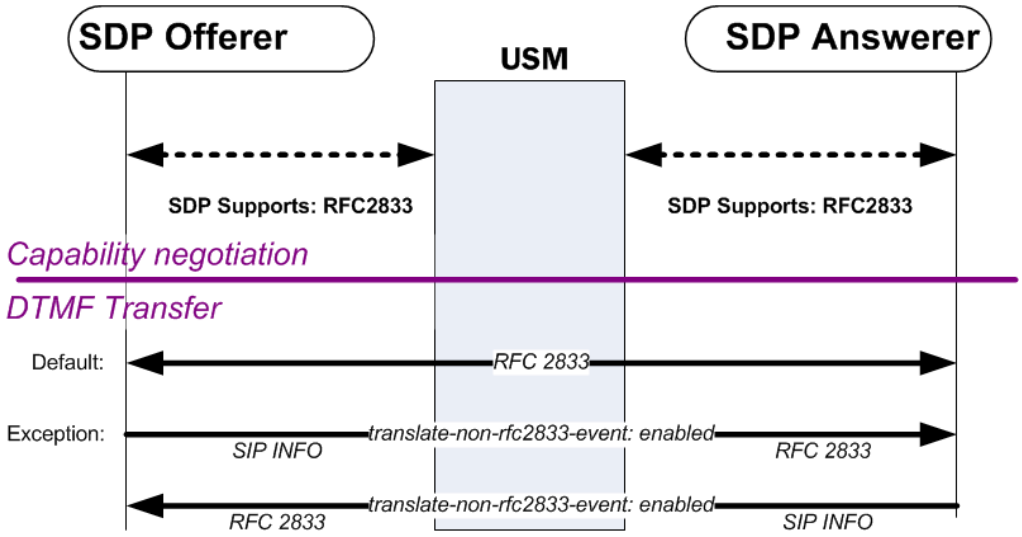
Identical Inband with Signaling DTMF Transfer Exception

Endpoints may send signaling-based DTMF indication (SIP INFO) to the Oracle USM at any time. Most of the time they are passed through the Oracle USM unchanged. You can enable an exception to this behavior forcing signaling messages to either RFC 2833 or DTMF audio tones depending on the call's DTMF transfer mode. The two parameters to enable their respective exceptions are located in the media manager configuration.

> **Note:**
>
> These behavior exceptions are only applicable when both sides of the call support and prefer the same DTMF transfer mode.

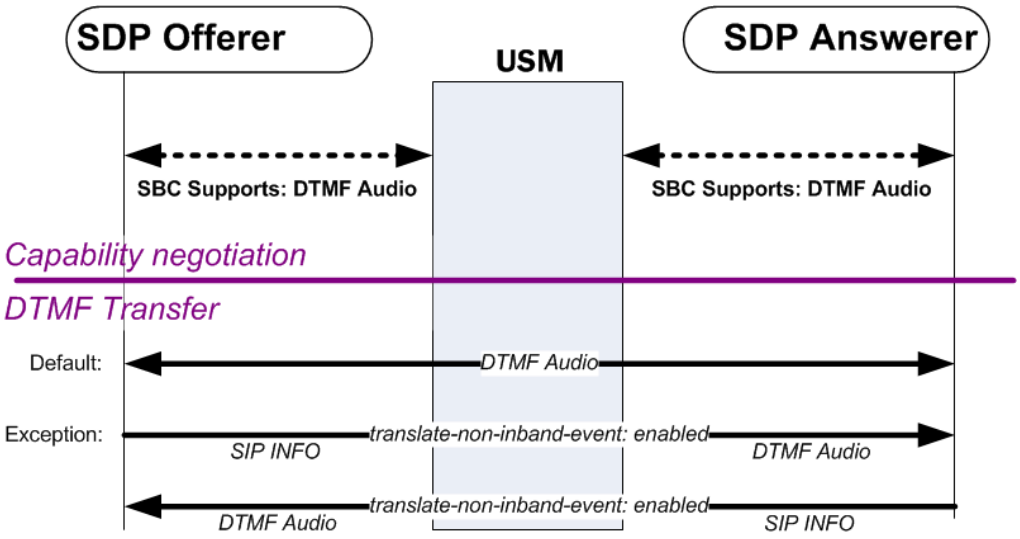
Override Preferred RFC 2833

When RFC 2833 is supported on both sides of the call, it is the preferred method of DTMF indication transport. To override this behavior, enable the translate non rfc 2833 parameter.



Override Preferred DTMF Audio

When DTMF audio tones are supported on both sides of the call, it is the preferred method of DTMF indication transport. To override this behavior, enable the translate non inband event parameter.



Note:

Enabling one or both of these exceptions can cost DTMF translation resources whether they occur in the network processors or on transcoding modules. The translate-non-inband-event exception is especially costing because it reserves transcoding for all calls that resolve to support for DTMFable to DTMFable codecs.

DTMF Transfer for Spiral Calls

A spiral call occurs when a call's signaling messages loop back through the Oracle USM. Most commonly the signaling path is from one UA, through the Oracle USM, to a call server, back through the Oracle USM, to another UA. The media path is from one UA, through the Oracle USM, to the other UA. For DTMF indication processing, only the call legs between the endpoints and Oracle USM are considered.

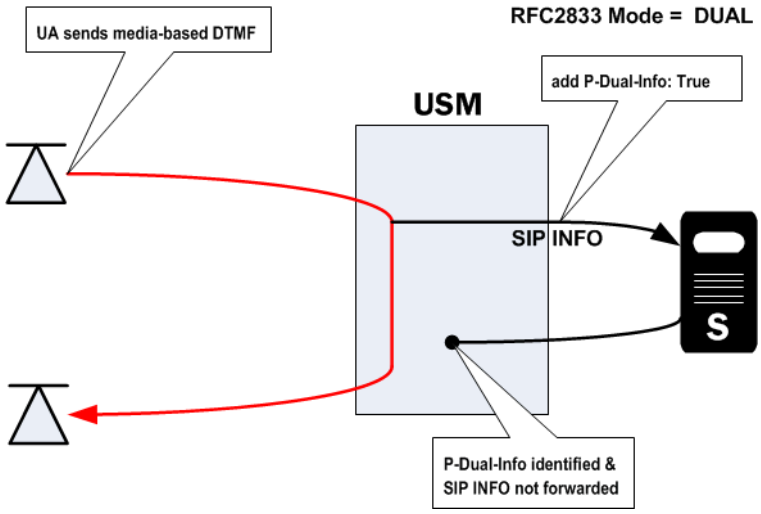
The Oracle USM evaluates that the signaling path to and from the call server terminates on the same IP address and port on the Oracle USM. This means that it's a spiral call. In addition, the media IP addresses and ports in the SDP indicate that the Oracle USM does not need to send the media into the call server's realm.

An issue occurs when DTMF indication is relayed either by RFC 2833 or DTMF audio tones for a spiral call. Since the DTMF indication is in the media path, the call server remains unaware of the signaling; no media-based DTMF indication digits will ever reach the call server. In order to include the call server in the DTMF-indication signaling, you should set the dtmf-in-audio and/or rfc-2833 mode used for the realm or signaling interface where the call server is to dual.

P-Dual-Info Header

The P-Dual-Info header is used in a spiral call scenario, when the call leg to (and from) the call server is set to dual. While the media portion of a spiral call goes from endpoint to endpoint through the Oracle USM, the signaling portion of the call loops through a call server.

The Oracle USM inserts a P-Dual-Info header into a SIP info message sent to the call server, which in turn forwards the SIP INFO message back to the Oracle USM. Seeing the P-Dual-Info header, the Oracle USM knows not to forward this SIP INFO message to the target endpoint because it would be a duplication of DTMF indication already sent to the endpoint in media format.



DTMF Transfer Hardware Processing

DTMF transfer processing, the conversion between two DTMF transfer types, occurs in either the transcoding NIU's Digital Signal Processors (DSPs) or the Network Processors (NPs). Understanding where the processing takes place is important to determine which subsystem uses extra processing load per conversion.

There are a few rules you can use to determine which subsystem performs the DTMF transfer processing:

- If audio transcoding is enabled for the call, DTMF transfer processing occurs in the transcoding modules.
- If DTMF audio tones are generated from RFC 2833 or from signaling messages (SIP INFO), DTMF transfer processing occurs in the transcoding modules.
- If the global translate non inband event parameter is enabled, DTMF transfer processing occurs in the transcoding modules.
- If signaling to RFC 2833 processing occurs in either direction of the call, and the previous 3 conditions are not valid, DTMF transfer processing occurs in the NPs.

DTMF Transfer Configuration

RFC 2833 Session Agent Configuration

Session agents, used as a way to classify and act on a subset of a signaling interface's traffic, also have **rfc2833-mode** and **rfc2833-payload** parameters. The configurations of these parameters overrides the configuration of the same-named parameters on the signaling interface

where the session agent resides. You can set the **rfc2833-mode** parameter to **none** for a session agent to revert to the parent signaling interface's two RFC 2833 settings.

ACLI Configuration and Instructions

This section explains how to configure the RFC 2833 mode on a signaling interface and on a session agent configured for that signaling interface. The session agent's configuration takes precedence over the signaling interface, unless the session agent's rfc2833-mode is set to none. In that case, the signaling interface's configuration is used for applicable traffic.

SIP Interface

To configure the RFC 2833 mode on a SIP interface:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-interface** and press Enter.

   ```
   ORACLE(session-router)# sip-interface
   ```

4. If you are adding support for this feature to a pre-existing SIP interface, then you must select the specific configuration instance, using the ACLI **select** command.
5. **rfc2833-mode**—Set this parameter to either **transparent**, **preferred**, or **dual** based upon the behavior your want for this SIP interface.
   - transparent—does not add RFC 2833 telephone-event into SDP if not present, and does not prefer.
   - preferred—adds RFC 2833 telephone-event media type into SDP and prefers to use this method for DTMF indication.
   - dual—adds RFC 2833 telephone-event media type into SDP and sends both SDP and signaling-based DTMF indications if possible.
6. **rfc2833-payload**—Set this parameter to the media-type value you wish to use when inserting RFC 2833 telephone-events into an SDP offer. 101 is the generally accepted media type for RFC 2833 telephone-events.
7. Save and activate your configuration.

Session Agent

Session agent RFC 2833 mode configurations override those on the signaling interface where they exit. The **none** parameter is used to defer to the signaling interface.

To configure the RFC 2833 mode on a session agent:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **session-agent** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# session-agent
   ORACLE(session-agent)#
   ```

4. Select the session agent where you want this feature.

   ```
   ORACLE(session-agent)# select 1
   ```

5. **rfc2833-mode**—Set this parameter to either **none, transparent**, **preferred**, or **dual** based upon the behavior your want for this SIP interface.
   - none—defaults to the behavior of the SIP interface for traffic that matches this session agent.
   - transparent—does not add RFC 2833 telephone-event into SDP if not present, and does not prefer.
   - preferred—adds RFC 2833 telephone-event media type into SDP and prefers to use this method for DTMF indication.
   - dual—adds RFC 2833 telephone-event media type into SDP and sends both SDP and signaling-based DTMF indications if possible.

6. **rfc2833-payload**—Set this parameter to the media-type value you wish to use when inserting RFC 2833 telephone-events into an SDP offer. 101 is the generally accepted media type for RFC 2833 telephone-events.

7. Save and activate your configuration.

Codec Policy

To configure a codec policy to support DTMF audio tones, as transcoded:

1. In Superuser mode, type configure terminal and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type media-manager and press Enter.

   ```
   configure# media-manager
   ```

3. Type **codec-policy** and press Enter.

   ```
   ORACLE(media-manager)# codec-policy
   ORACLE(codec-policy)#
   ```

4. If you are adding support for this feature to a pre-existing configuration, then you must select the specific configuration instance, using the ACLI **select** command.

   ```
   ORACLE(codec-policy)# select
   <name>:
   1: private
   2: public
   selection:1
   ORACLE(codec-policy)#
   ```

5. **dtmf-in-audio**—Set this parameter to **disabled**, **preferred**, or **dual** based upon how the Oracle USM should support the conversion of signaling messages or RFC 2833 to DTMF Audio tones in the realm where this codec policy is active.
   - disabled—does not support DTMF audio tones as transcoded in this realm.
   - preferred—supports DTMF audio tones as transcoded in this realm.

- dual—supports both transcoded DTMF audio tones and signaling-based DTMF indications if possible.

6. Save and activate your configuration.

Translate Non2833 Event Behavior

To configure the exceptional behavior:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring this feature.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **translate-non-rfc2833-event**—Set this parameter to enabled to use non-default behavior described in Override Preferred RFC 2833.
5. **translate-non-inband-event**—Set this parameter to enabled to use non-default behavior described in Override Preferred DTMF Audio.
6. Save and activate your configuration.

P-dual-info Header Appearance

Customizing the P-Dual-Info header is performed globally from the sip config.

To configure how the P-dual-info header appears:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **session-router** and press Enter to access the system-level configuration elements.

   ```
   ORACLE(configure)# session-router
   ```

3. Type **sip-config** and press Enter. The system prompt changes to let you know that you can begin configuring individual parameters.

   ```
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

4. Type options followed by a Space.
5. After the Space, type the P-Dual-Info header information in the following format:

   ```
   +dual-info="<header-name>"
   ```

   For example:

   ```
   ORACLE(sip-config)# options dual-info=P-Dual-Info
   ```

6. Save your work using the ACLI **done** command.

RFC 2833 Customization

RTP Timestamp

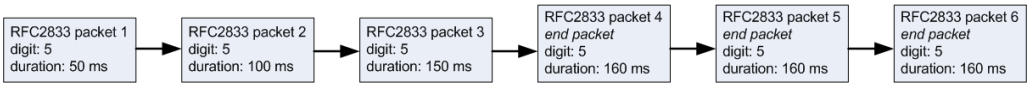
As a media flow with injected RFC 2833 telephone-event packets exits the Oracle USM, the newly generated RTP packets are timestamped in one of two ways. If RFC 2833 generation is preformed in the NPs, the default method of creating the timestamp for a generated RFC 2833 packet is to use the previous RTP packet's timestamp and add 1.

Alternatively the Oracle USM, can estimate the actual time that the injected RFC 2833 telephone-event packet will leave the system and use that. This method tags the injected DTMF indication packet more accurately than the than previous packet + 1 method. As an additional bonus, the packet's checksum is regenerated. This alternate timestamp creation behavior is configured by setting the **rfc2833-timestamp** parameter to enabled.

When RFC 2833 telephone-event generation is preformed by the transcoding modules, the packets' timestamps are the set to the time that the packets leave the Oracle USM,.

RFC 2833 telephone-event duration intervals

If an incoming SIP INFO message's DTMF indication duration is unspecified, the Oracle USM, uses a default 250 ms duration for the generated RFC 2833 telephone-event. Otherwise, the SIP INFO's specified event duration is used. RFC 2833 telephone-event packets are still generated at 50 ms intervals upon egress. At the conclusion of the DTMF indication, the three end-event packets are sent. The packet arrangement when the user presses the digit 5 for 160ms, with the default 50ms interval follows:



When either no DTMF event duration is specified, or the event duration is less than the 50ms default minimum, you can set the default RFC 2833 telephone-event duration using **default-2833-duration** parameter in the media manager configuration. This is the value that the Oracle USM, uses for the duration of a telephone event when none is specified in the incoming message. The **default-2833-duration**'s valid range is 50-5000ms. The Oracle USM, also uses this configured value when it receives a SIP INFO message with a duration less than the minimum signal duration.

You can configure the minimum duration at which RFC 2833 telephone-events are generated by the Oracle USM, using the **min-signal-duration** option in the media manager configuration, thus changing the lower threshold of the **default-2833-duration** parameter from 50 ms to your own value. If the duration the Oracle USM, receives is less than the threshold, it uses the value configured in the **default-2833-duration** parameter.

> **Note:**
>
> Timestamp changes and duration changes only take effect when the 2833 timestamp (rfc-2833-timestamp) is enabled in the media manager configuration. If you enable the rfc-2833-timestamp parameter, but do not configure the default-2833-duration parameter, the default-2833-duration parameter defaults to 100 ms.

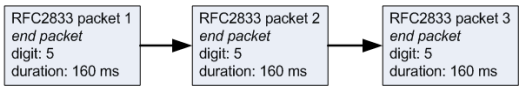
RFC 2833 End Packets

When the Oracle USM, generates RFC 2833 telephone-event packets, they are forwarded from the egress interface every 50 ms by default. Each packet includes the digit and the running total of time the digit is held. Thus DTMF digits and events are sent incrementally to avoid having the receiver wait for the completion of the event.

At the conclusion of the signaled event, three end packets stating the total event time are sent. This redundancy compensates for RTP being an unreliable transport protocol.

You can configure your Oracle USM, to generate either the entire start-interim-end RFC 2833 packet sequence or only the last three end 2833 packets for non-signaled digit events using the **rfc2833-end-pkts-only-for-non-sig** parameter. If the parameter were enabled, the RFC 2833 telephone-event packets for the same event would appear are represented by the following graphic.

rfc2833-end-pkts-only-for-non-sig = *enabled*



ACLI Instructions and Examples

To configure RFC 2833 customization:

1. In Superuser mode, type **configure terminal** and press Enter.

   ```
   ORACLE# configure terminal
   ```

2. Type **media-manager** and press Enter to access the media-level configuration elements.

   ```
   ORACLE(configure)# media-manager
   ORACLE(media-manager)#
   ```

3. Type **media-manager** and press Enter to begin configuring this feature.

   ```
   ORACLE(media-manager)# media-manager
   ORACLE(media-manager-config)#
   ```

4. **rfc2833-timestamp**—Set this parameter to enabled to use the estimated real departure timestamp on an injected RFC 2833 telephone-event packet.
5. **default-2833-duration**—Set this parameter to the default value you wish to use when the Oracle USM, creates or generates DTMF-indication messages.
6. **rfc2833-end-pkts-only-for-non-sig**—Set this parameter to enabled for the Oracle USM, to only send the three RFC 2833 telephone-event end-packets to indicate a total event duration, rather than the running total from time=0.
7. **options**—Set the options parameter by typing **options**, a Space, the option name **min-signal-duration=x** (where **x** is the value in milliseconds you want to use for the threshold) with a plus sign in front of it. Then press Enter.
8. Save and activate your configuration.

21
Tunneled Services Control Function

TSM/TSCF Overview

Tunnel Session Management (TSM) is based upon an existing 3rd Generation Partnership Project (3GPP) Technical Requirement, TR 33.8de V0.1.3 (2012-05) that seeks to define a standardized approach for overcoming non-IMS aware firewall issues. Within the 3GPP, TSM is referred to as Tunneled Services Control Function or TSCF. Oracle uses the 3GPP terminology: TSCF.

TSCF improves firewall traversal for over-the-top (OTT) Voice-over-IP (VoIP) applications, and reduces the dependency on SIP/TLS and SRTP by encrypting access-side VoIP within standardized Virtual Private Network (VPN) tunnels. As calls or sessions traverse a TSCF tunnel, the Oracle Communications Tunneled Session Controller forwards all SIP and RTP traffic from within the TSCF tunnel to appropriate servers or gateways within the secure network core. Operating in a TSM topology, the SBC provides exceptional tunnel performance and capacity, as well as optional high availability (HA), DoS protection and tunnel redundancy that improves audio quality in lossy networks.

As implemented by Oracle Communications, TSM/TSCF terminology includes:

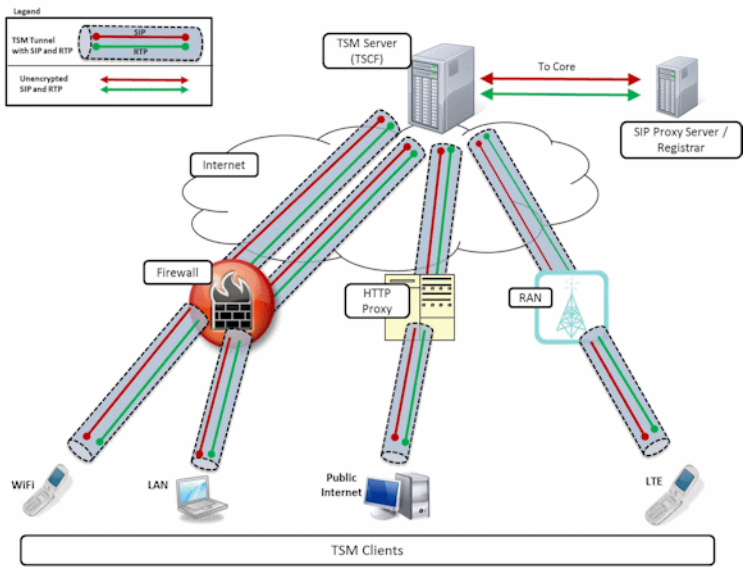
- TSC client—Sometimes referred to as a Tunneled Service Element (TSE). It facilitates TSCF tunnel creation and management within client applications residing on network elements.
- TSC server—The SBC that is used to terminate TSC tunnels.
- Tunnel Session Management (TSM)— The overall solution that covers both the TSC Server and TSC Client.
- TSCF (Tunneled Services Control Function)—The actual function that runs on the SBC. TSCF is often used in the naming of the ACLI configuration and management objects.

A TSC client runs within client applications residing on network elements — for example: workstations, laptops, tablets and mobile devices (Android, iPhone, Windows, or iPad). Oracle provides a software development kit (SDK) that facilitates TSCF tunnel creation and management. Refer to the companion document, the *TSE SDK Guide*, or to the SDK on-line Help system for information on client-side programming, and to access reference applications that provide client-specific coding examples for supported operating systems.

To deploy TSC clients, customers and 3rd party Independent Software Vendors (ISVs) need to link the open source TSC client libraries with their applications, which can then initiate and establish SSL VPNs (TLS or DTLS) to the TSC server.

In contrast to a TSC client, a TSC server resides on the Oracle Communications Session Boarder Controller (OCSBC), and provides server-side functions that accept, establish, and manage tunnel connections between remote TSC-client-enabled applications and core Proxy-Call Session Control Function and SIP proxies (P-CSCFs/SIP proxies). The following illustration shows various ways in which TSC clients can be deployed across different network segments (WiFi, LAN, WAN, LTE) and interface with common network elements (firewalls, HTTP proxies, and radio access network equipment).

Figure 21-1 TSM /TSCF Topologies



TSCF Entitlements

In order to use correct feature of TSCF, the below entitlement commands need to be used to apply correct setup.

If this is your first time use the TSCF entitlement, you first need to setup product as below.

```
ORACLE# setup entitlements
Product not initialized; Please use 'setup product'

ORACLE# setup product

----------------------------------------------------------------
WARNING:
Alteration of product alone or in conjunction with entitlement
changes will not be complete until system reboot

Last Modified
----------------------------------------------------------------
1 : Product       : Uninitialized

Enter 1 to modify, d' to display, 's' to save, 'q' to exit. [s]: 1

  Product
    1 - Session Border Controller
    2 - Session Router - Session Stateful
    3 - Session Router - Transaction Stateful
    4 - Peering Session Border Controller
  Enter choice     :

  Select either 1 or 4 above depending other features you are using.
```

Use the **show entitlements** command to display your TSCF enabled status.

```
ORACLE# show entitlements
Provisioned Entitlements:
```

```
------------------------
Session Border Controller Base    : enabled
Session Capacity                  : 32000            <---- This setup also
  Accounting                      :
enabled                                                          used for
TSCF
  IPv4 - IPv6 Interworking        : enabled
  IWF (SIP-H323)                  : enabled
  Load Balancing                  : enabled
  Policy Server                   : enabled
  Quality of Service              : enabled
  Routing                         : enabled
  SIPREC Session Recording        : enabled
Admin Security with ACP/NNC       :
IMS-AKA Endpoints                 : 0
IPSec Trunking Sessions           : 0
MSRP B2BUA Sessions               : 0
SRTP Sessions                     : 0
TSCF Tunnels                      : 10000             <---- Non-zero to


enable TSCF



entitlement.
```

Use **setup entitlements** to change entitlements for your TSCF enabled status, as below.

```
ORACLE# setup entitlements
 ----------------------------------------------------
Entitlements for Session Border Controller
Last Modified: 2015-05-19 22:07:05
----------------------------------------------------
 1 : Session Capacity                      : 32000
 2 :   Accounting                          : enabled
 3 :   IPv4 - IPv6 Interworking            : enabled
 4 :   IWF (SIP-H323)                      : enabled
 5 :   Load Balancing                      : enabled
 6 :   Policy Server                       : enabled
 7 :   Quality of Service                  : enabled
 8 :   Routing                             : enabled
 9 :   SIPREC Session Recording            : enabled
 10: Admin Security with ACP/NNC           :
 11: IMS-AKA Endpoints                     : 0
 12: IPSec Trunking Sessions               : 0
 13: MSRP B2BUA Sessions                   : 0
 14: SRTP Sessions                         : 0
 15: TSCF Tunnels                          : 10000

Enter 1 - 15 to modify, d' to display, 's' to save, 'q' to exit. [s]:
```

To disable TSCF entitlement feature, set TSCF Tunnels to zero.

To enable TSCF Decomposed feature only, set above Session Capacity to zero.

The above setting is for TSCF Coupling or TSCF SBC Enabled status.

License Requirements

TSCF availability is dependant on the acquisition and installation of a TSCF license.

The TSCF license supports distinct tunnel tiers (counts) as follows: 1K, 5K 10K, 20K, 30K, 40K, 50K, 60K, 70K, 80K, 90K, 100K, 110K, 120K, 130K, 140K, 150K, 160K, 170K, 180K, 190K, 200K.

TSCF also requires the TLS license, and standard protocol-based licenses (for example, SIP) based on network traffic requirements. The Denial of Service detection/prevention capability described in Denial of Service, requires a DoS license.

Use the **show features** ACLI command to display a list of installed licenses. The resulting display will resemble the following example that displays a superset of available licenses.

```
ORACLE# show features
Total session capacity: 32000
TSCF tunnel capacity : 200000
Enabled features:
        SIP, H323, IWF, QOS, ACP, Routing,
        Load Balancing, Accounting, High Availability,
        PAC, LI, External BW Mgmt, TLS, Software TLS,
        External CLF Mgmt, External Policy Services, ENUM, H248,
        H248 SCF, H248 BGF, NSEP RPH, LI Debug,
        Session Replication for Recording, Transcode Codec AMR,
        Transcode Codec EVRC, DoS, IKE, IPv4-v6 Interworking, RTSP,
        IDS, Transcode Codec EVRCB, Software PCOM, Security Gateway,
        SIP Authorization/Authentication,
        Database Registrar (0 contacts), IDS Advanced,
        SLB (2000000 endpoints), Allow Unsigned SPL files,
        Session Recording, Policy Director,
        TSCF (200000 TSCF tunnels), Transcode Codec AMR-WB, CX
ORACLE#
```

Deployment Models

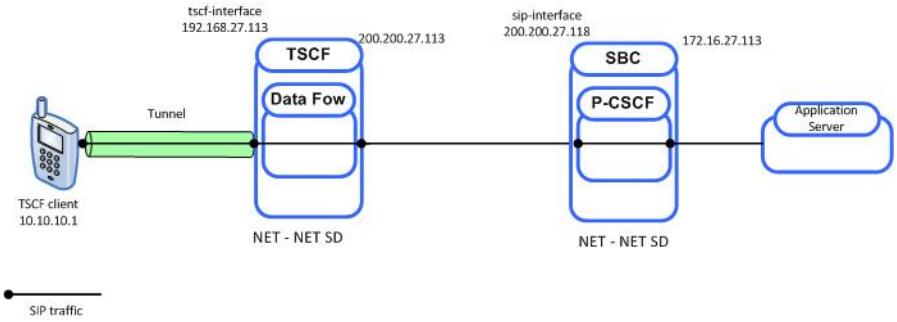
Based on local network topology and requirements, TSM service can be deployed in two ways:

- Decomposed model
- Combined model

Decomposed Model

The decomposed model provides TSC functionality on an Oracle server (for example, a Multiprotocol Security Gateway or a generic SBC). Regardless of the supporting platform, the TSC server acts as a pass through server, using a data-flow (refer to TSCF Data Flow Configuration) to forward all tunnel traffic to a pre-determined IP address. In the simplest topology, this address would access a gateway providing the protocol support required to route the previously tunneled traffic to its ultimate destination. Alternatively, the data-flow could direct tunneled traffic to another SBC that provides P-CSCF (Proxy-Call Session Control Function) services as shown in the following illustration.

Figure 21-2 Decomposed Model



Within this decomposed TSC server:

- 192.168.27.113 provides a TSCF interface to the access realm (refer to TSCF Interface Configuration for procedural details)
- The TSCF interface has an assigned address pool (refer to TSCF Address Pool Configuration for procedural details) that provides a contiguous range of tunnel addresses (for example, 10.10.10.1 available to TSC clients
- The TSCF address pool has an assigned data-flow (refer to TSCF Data Flow Configuration for procedural details) that specifies a static route to the core realm
- 200.200.27.113 provides a SIP interface that provides transit to the core realm

Within the P-CSCF server:

- 200.200.27.18 provides a SIP interface serving the core realm
- 172.16.27.113 provides an interface to a SIP application server

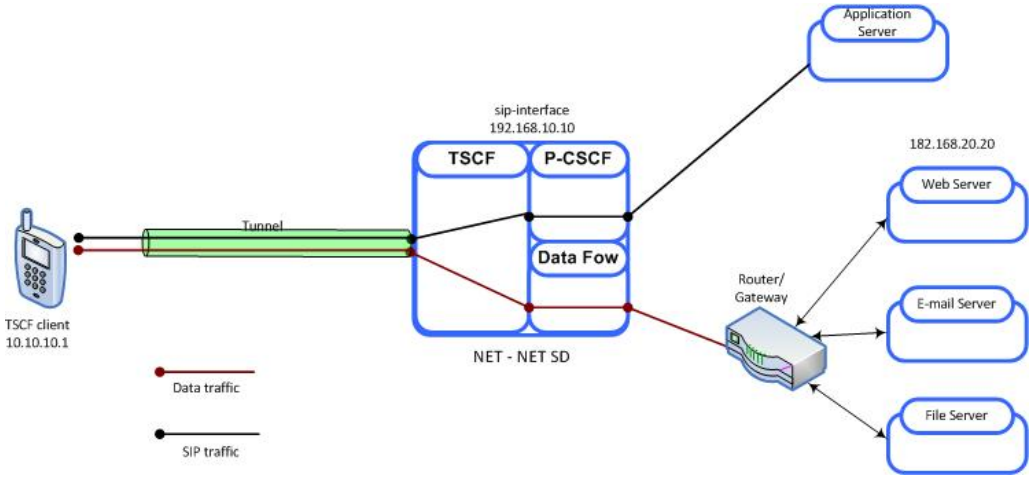
Decomposed packet flow can be summarized as follows:

1. The TSC client connects to the TSC server at 192.168.27.113
2. Assuming client authentication succeeds, the TSC server accepts connection and assigns the TSC client a tunnel address (10.10.0.1)
3. Using the tunnel, the TSC client sends a SIP INVITE from 10.10.0.1 to a remote peer
4. Using the data-flow, the TSC server forwards the INVITE to the core realm (200.200.27.118)
5. The P-CSCF forwards the INVITE to the SIP application server

Combined Model

The combined model involves deploying both the TSCF and P-CSCF on the same Oracle SBC. With this model, the TSCF can be configured to pass SIP traffic to the co-located P-CSCF (thus providing standard SIP processing), and to use a data-flow to pass all non-SIP traffic to a designated destination, most often a gateway that provides the protocol support required to route the previously tunneled data traffic to its ultimate destination.

Figure 21-3 Combined Model - Standard Processing



Within this combined TSC server:

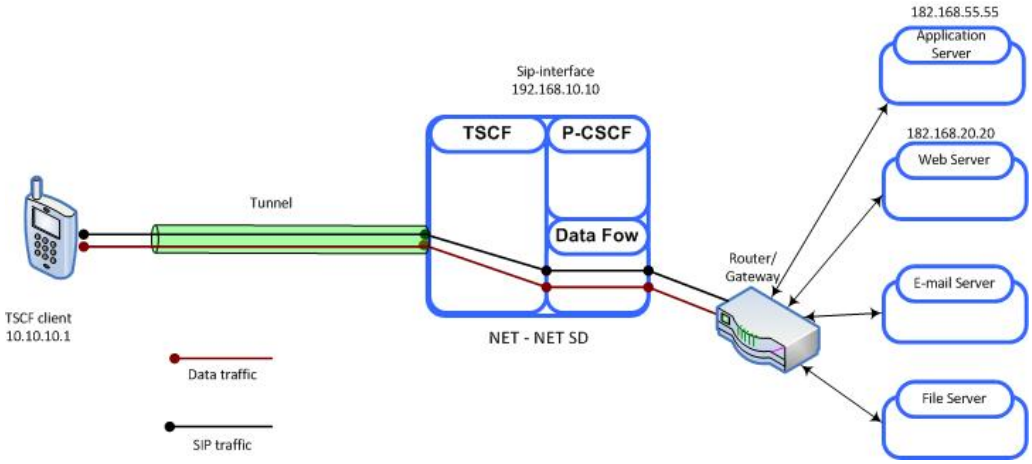
- 192.168.10.10 (a SIP interface) provides a TSCF interface to the access realm (refer to TSCF Interface Configuration for procedural details)
- SIP signalling traffic received from a tunnel endpoint (and explicitly addressed to 192.168.10.10) is directed to an appropriate P-CSCF process for forwarding to a generic or specialized SIP application server
- The TSCF interface has an assigned address pool (refer to TSCF Address Pool Configuration for procedural details) that provides a contiguous range of tunnel addresses (for example, 10.10.1) available to TSC clients
- The TSCF address pool has an assigned data-flow (refer to TSCF Data Flow Configuration for procedural details) that specifies a static route to a router/gateway; any data traffic received from a tunnel endpoint is passed through to the gateway

The following figure illustrates traffic pass-through within the Combined model.

Within this combined TSC server:

- 192.168.10.10 (a SIP interface) provides a TSCF interface to the access realm (refer to TSCF Interface Configuration for procedural details)
- The tunnel client sends SIP messages to 182.168.55.55 (a SIP interface not resident on the Oracle SBC)
- In this case, all traffic is subject to the data-flow, which directs both SIP signalling and media streams to a gateway for routing to the destination IP address

Figure 21-4 Combined Model - Pass-Through Processing

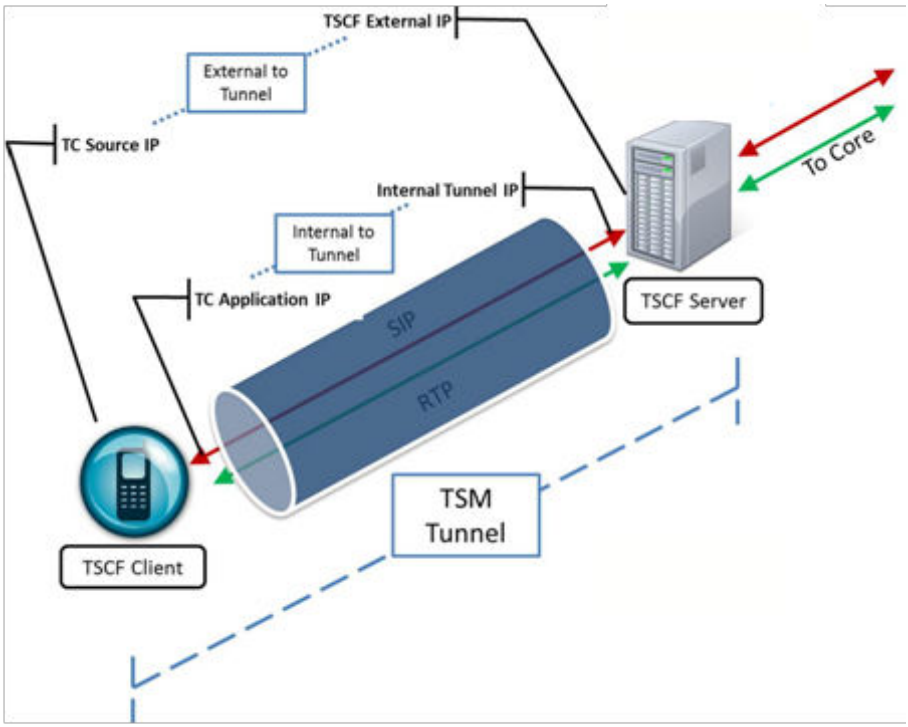


Tunnel Establishment

Tunnel creation is initiated by the client application that first establishes a TLS and/or DTLS tunnel between the TSC client and the TSC server. Tunnel creation is completed with a single exchange of configuration data accomplished by a TSC client request and TSC server response. Refer to the *TSE SDK Guide* for detailed information on tunnel set-up procedures.

The following illustration briefly explains the IP addresses used during TSM tunnel establishment and operation.

Figure 21-5 TSM Tunnel Address Structure



- TSCF External IP—This public IP address is visible to any endpoint; TSC client requests for tunnel establishment are directed to this address (refer to TSCF Interface Configuration for procedural details).
- TC source IP—This public IP address identifies either the source address of the TSC client in its respective access network, or the IP address of an intervening proxy, firewall, or NAT device.
- Internal Tunnel IP—This private address will be assigned to the TSC client (assuming authentication is successful) from a configured pool of IP addresses on the TSC server. Refer to TSCF Address Pool Configuration for procedural details.
- TC Application IP—This private IP address identifies a specific application (SIP/RTP/etc.) at the TSC client site.

All packets between the client application on the TSC client and the TSC server are comprised of inner and outer parts separated by a TLS header. All data after the TLS header is encrypted.

- Outer headers contain TSC client and TSC server addresses
- Inner headers contain application and P-CSCF addresses
- Payload packets carry application data between the TSC client and the TSC server
- Control payloads support tunnel management and configuration activities such as the assignment of a TSC client inner IP address, and the exchange of keepalive messages

The following two illustrations depict simplified packet formats.

Figure 21-6 Data Traffic Packet Structure

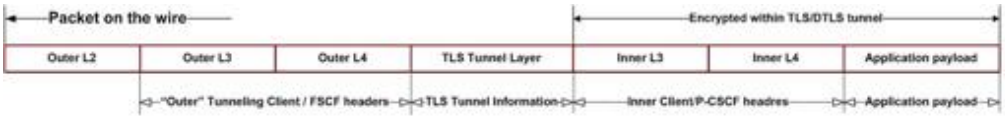
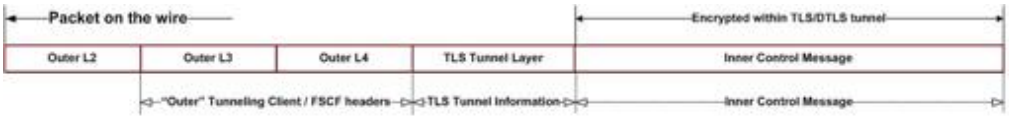
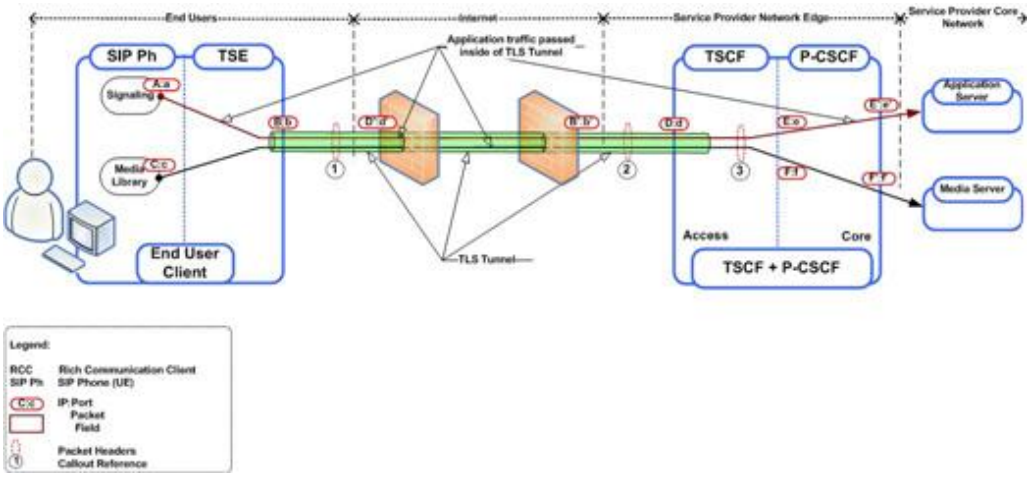


Figure 21-7 Control Traffic Packet Structure



The following illustrations provide a more detailed view of packet structure and addressing. The first figure provides a network reference model.

Figure 21-8 Reference Model



These two figures show a client-initiated and a server-initiated packet with sample addresses.

Figure 21-9 TSC Client to TSC Server

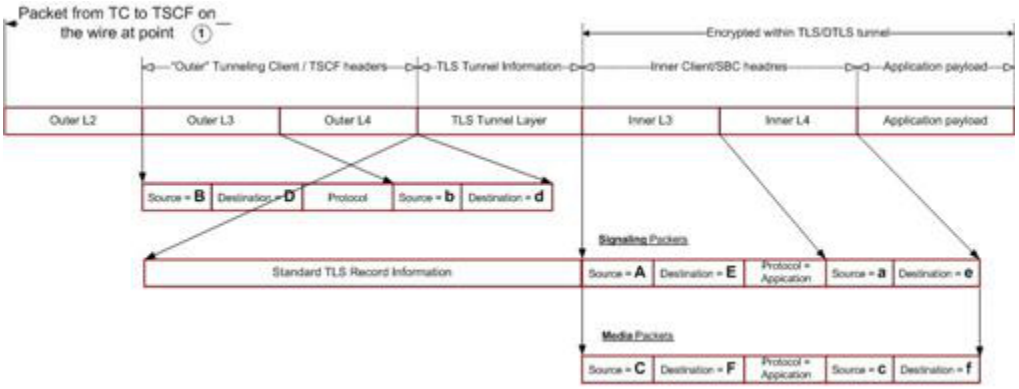
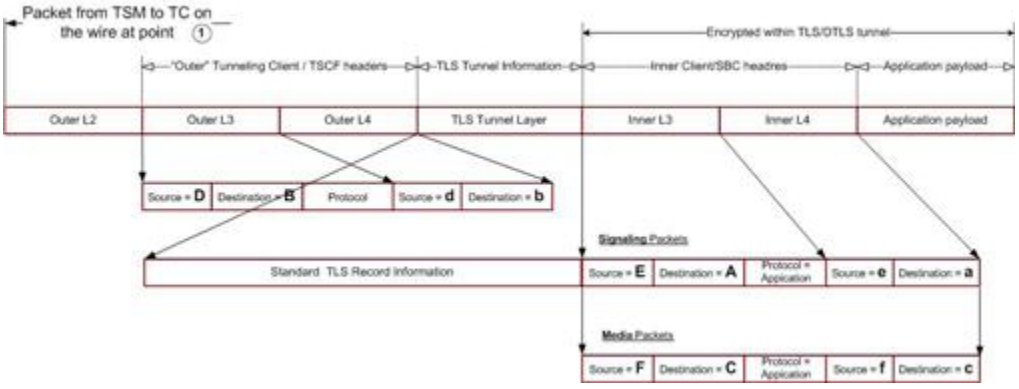


Figure 21-10 TSC Server to TSC Client



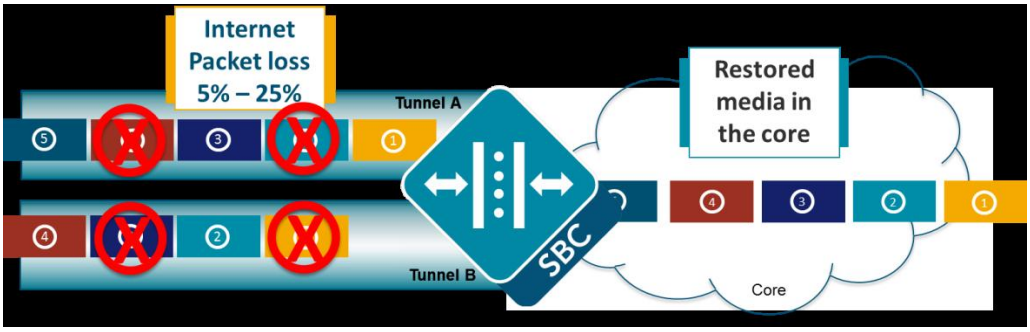
Tunnel Redundancy

The ability to establish parallel, redundant tunnels improves media quality under adverse network packet loss conditions (defined as packet loss within the 5% to 25% range). As the name implies, tunnel redundancy creates secondary TSCF tunnels that replicate signaling and media transport between the TSC client and the TSC server.

Tunnel redundancy is one of the assigned services, along with ddt, server-keepalive, and sip, that can be enabled or disabled on a TSCF interface; by default, tunnel redundancy is disabled. This service is enabled by including the string redundancy in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details).

With tunnel redundancy enabled, as packets traverse networks and encounter packet loss and drop, the TSC client and TSC server choose the most available and timely packet from the duplicative tunnels. Within redundant tunnels, packets are slightly offset (as shown in the following figure), so as a gateway or other intervening network elements drop packets, the same packet that was dropped in one tunnel remains in another tunnel. When the packets reach their destination in the core, or on the client, the media stream is reformed as if packet loss never happened.

Figure 21-11 Tunnel Redundancy



Tunnel redundancy works in concert with RTP redundancy and packet loss concealment algorithms that are often implemented by advanced variable bit rate codecs such as SPEEX, SILK or iSAC. Like RTP redundancy, defined in RFC 6354, Forward Shifted RTP Redundancy Payload Support, tunnel redundancy does increase bandwidth usage by replicating signaling and media but also maximizes the chances of packet delivery. In addition to packet loss, tunnel redundancy can also mitigate the effects of jitter (variable packet delay), a common network impairment that can result in significant speech quality degradation.

A TSC client can initiate tunnel redundancy at call establishment or, in response to current network conditions, at any time within an established call session. Depending on the transport protocol (TCP or UDP) a TSC client can request one of three available tunnel redundancy modes:

- TLS/TCP Load Balancing
- TLS Fan-Out
- DTLS/UDP Redundancy

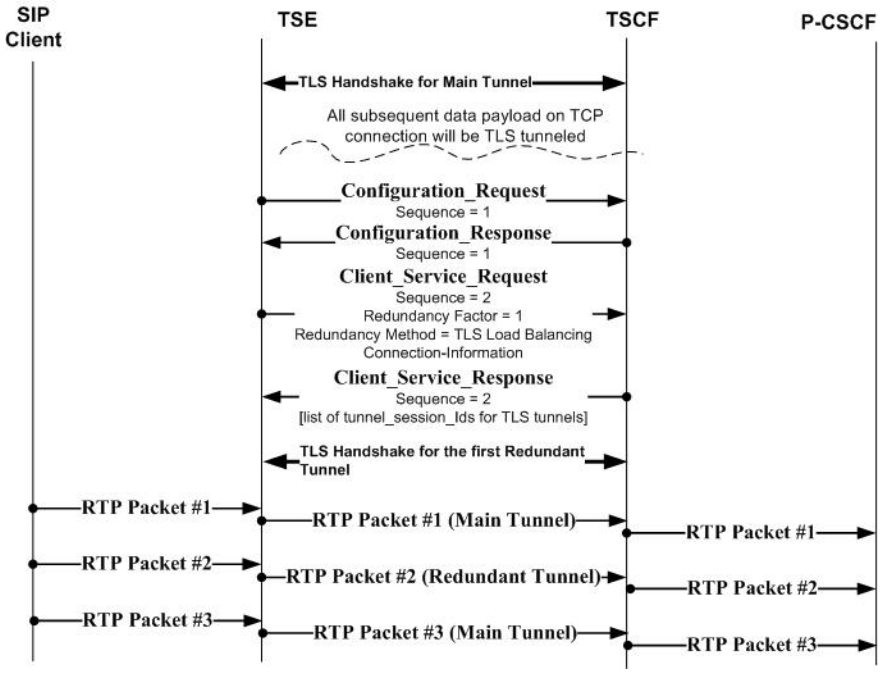
TLS/TCP Load Balancing

TLS/TCP load balancing mode is based on parallel, redundant tunnels. Both the TSC client and the TSC server send RTC encapsulated packets on one tunnel at a time. Each subsequent packet is sent on the next tunnel in a circular fashion. The following illustration shows packet transmission over time with packet transmission balanced over a main tunnel and two redundant tunnels.

Figure 21-12 Redundancy — TLS/TCP Load Balancing

This illustration provides a sample message flow for TLS/TCP load balancing negotiation.

Figure 21-13 TLS/TCP Load Balancing Message Flow

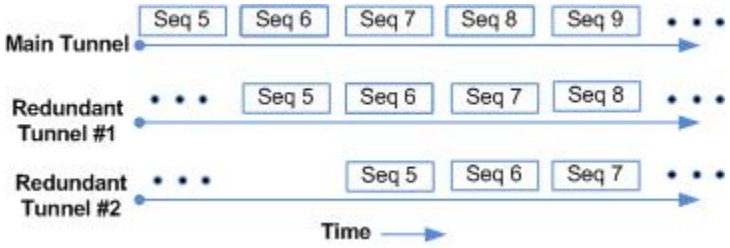


TLS TCP Fan-Out

TLS/TCP fan-out mode is also based on redundant parallel tunnels. Both the TSC client and the TSC server send RTC encapsulated packets on the main tunnel. Additionally both the client and server send the same packet on each redundant tunnel in a time-staggered fashion. The

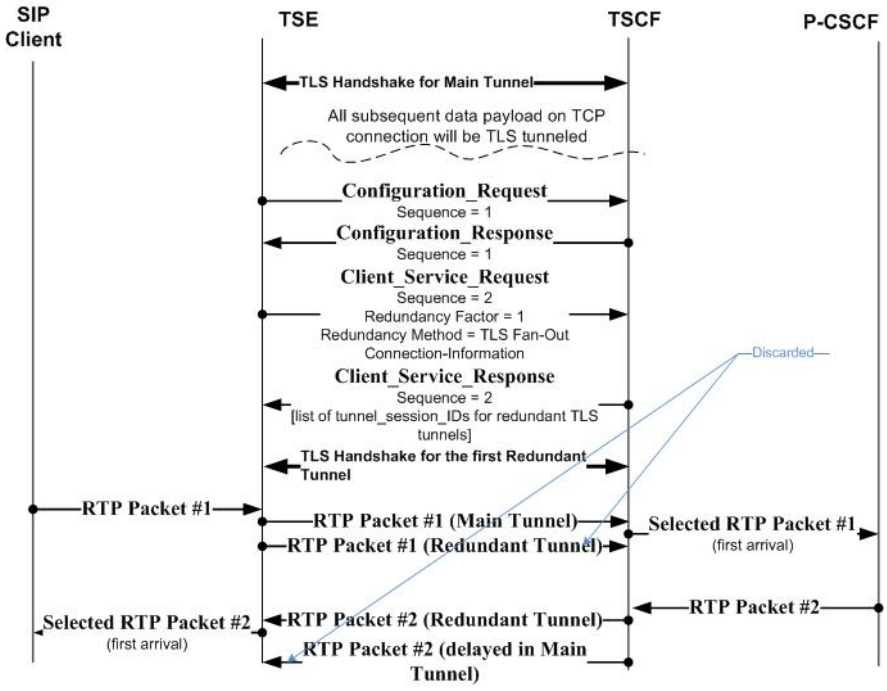
following illustration shows packet fan-out over time with a main tunnel and two redundant tunnels.

Figure 21-14 Redundancy — TLS/TCP Fan-Out Load Balancing



This illustration provides a sample message flow for TLS/TCP fan-out.

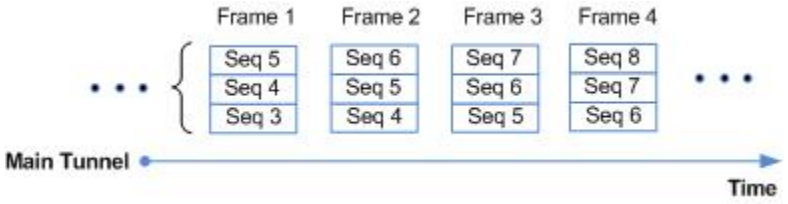
Figure 21-15 TLS/TCP Fan-Out Load Balancing Message Flow



DTLS UDP Redundancy

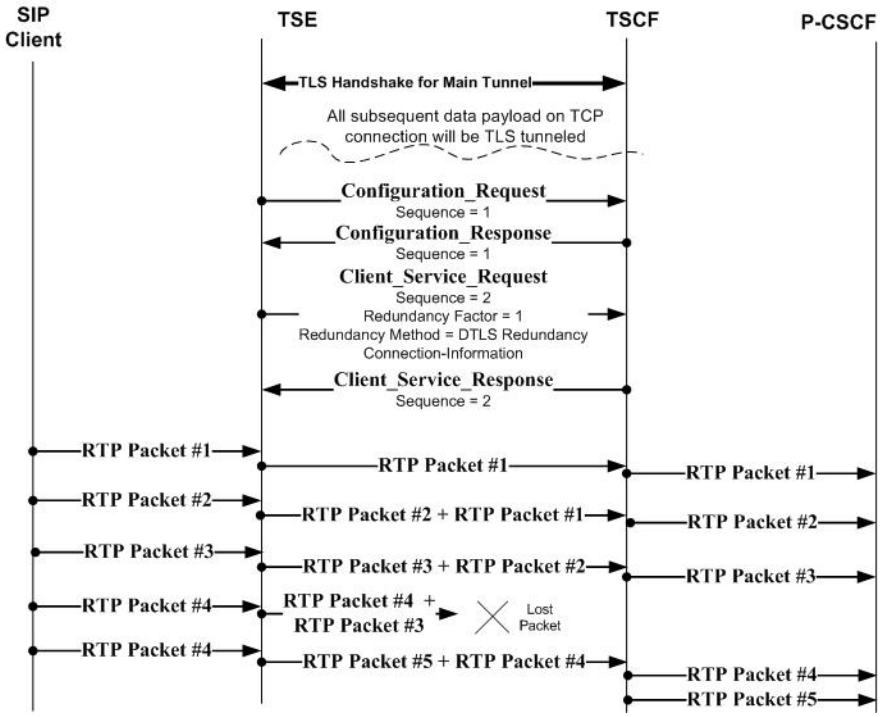
DTLS/UDP redundancy mode, unlike TLS/TCP load balancing and TLS/TCP redundancy modes, does not establish redundant parallel tunnels between the TSC client and TSC server. Instead, both the client and server exchange frames of RTC encapsulated packets. Each frame contains a new packet and some number of sequential, previously sent packets. The following illustration shows DTLS/UDP redundancy over time.

Figure 21-16 Redundancy — DTLS/UDP Redundancy



This illustration provides a sample message flow for DTLS/UDP redundancy.

Figure 21-17 DTLS/IDP Load Balancing Message Flow



Denial of Service

Denial of Service (DoS) protection is a licensed capability that adds preliminary access control and bandwidth control for tunneled traffic.

Because tunneled traffic is confined to a small number of well-defined ETC-based ports, traffic processing can be streamlined. Incoming traffic is directed to one of two pipes, each supported by dedicated queues, for transmission toward the network core. A larger pipe provides maximal bandwidth for trusted tunnel endpoints, while a smaller pipe provides minimal bandwidth for untrusted tunnel endpoints.

Tunnels can be promoted to the trusted level, or demoted to the untrusted level.

Initially all incoming traffic is treated as untrusted and directed to the untrusted pipe. When the TLS handshake is successfully completed (meaning that the two peers have mutually

authenticated) the tunnel transitions to the trusted state. The tunnel remains in the trusted state until the expiration of the tunnel persistence timer, as described in TSCF Global Configuration.

Server-Initiated Keepalive

For some TSC client devices, such as IPhones or iPads, operating system constraints can prevent the transmission of timely keepalive messages required to refresh an existing TSCF tunnel connection or to maintain current NAT bindings. To address this problem the TSC server can be configured to issue periodic server-initiated keepalive (SIK) messages to clients who require such service.

SIK is one of the assigned services, along with ddt, redundancy, and sip, that can be enabled or disabled on a TSCF interface; by default, SIK is disabled. SIK is enabled by including the string server-keepalive in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details).

SIK services are provided only in response to a TSC client request. The client requests SIK services during the configuration process. Upon receiving such a request, the TSC server checks the status of SIK on the interface on which the service request was received. If SIK is enabled, the request is accepted; if SIK is disabled (the default state), the request is denied.

Dynamic Datagram Tunneling

Dynamic datagram tunneling (DDT) provides parallel stream-based (TCP/TLS) and datagram-based (UDP/DTLS) tunnels to address problems/challenges in delivering Real-Time Protocol (RTP) media over a connection-oriented, reliable TCP transport. With DDT enabled, the TSC client and TSC server negotiate a single secure, persistent, stream-based tunnel, used for SIP signaling and tunnel maintenance, as well as a series of dynamic datagram-based tunnels used for RTP media transport.

With DDT enabled the TSC server assigns the same inner tunnel address to both the stream-based and datagram-based tunnels. The initial datagram-based tunnel provides a template for subsequent tunnels created to service pending RTP packets. Each tunnel is configured in exactly the same manner as the initial tunnel, save for the tunnel ID, which is unique for each tunnel instance.

Dynamic datagram tunneling is one of the assigned services, along with redundancy, server-keepalive, and sip, that can be enabled or disabled on a TSCF interface; by default, dynamic datagram tunneling is disabled. The service is enabled by including the string ddt in the comma-separated assigned-services list (refer to TSCF Interface Configuration for procedural details). DDT configuration has some unique requirements as listed below:

- **assigned-services** MUST be set to sip,ddt
- a TLS/TCP port must be configured on the TSCF interface
- a DTLS/UDP port, using the same IP address and port number as assigned to the TLS/TCP port must be configured on the same TSCF interface

Dynamic datagram tunneling service is provided only in response to a TSC client request. The client requests this service during the configuration process. Upon receiving such a request, the TSC server checks the status of dynamic datagram tunneling on the interface on which the service request was received. If the service is enabled, the request is accepted; if the service is disabled (the default state), the request is denied.

Tunnel Restoration and High Availability

Datagram Transport Layer Security (DTLS) and Dynamic Datagram Tunnels (DDT) support disruption-free traffic flow during High Availability (HA) failover.

During normal operations, the active member of the HA pair synchronizes certain DTLS tunnel details with its stand-by partner to include internal IP addresses, tunnel identifiers, cryptographic materials, and additional DTLS context information. DTLS tunnels carry the media stream. Upon synchronization, the stand-by member populates the DTLS stack with the received data, and creates DTLS tunnels in a quasi-active state — such tunnels do not actually process packets. If and when HA handover occurs, the presence of the traffic-ready DTLS tunnels mitigates against disruption of the media stream.

When an HA pair experiences a failover (or in the event of any disconnect), TSC clients have a grace period (defined by the **tunnel-persistence-time** parameter) to recognize the failure and re-connect using the same tunnel. If clients react in a timely fashion, the newly active system allocates the same TSCF tunnel identifier and IP address.

A similar, although not identical approach, is used in DDT environments.

DDT tunnel restoration proceeds as follows:

1. The active member updates the DTLS tunnels, which carry the media streams with augmented contextual information as described above.
2. In the event of HA failover, the TLS tunnels, which carry control traffic and signalling, continue in the persistent state, and remain in that state until:
   - the stand-by transitions to the active role, and
   - the client re-connects

   Only after these two conditions are met does the TLS tunnel transition to the active state.
3. If the TSC client requests restoration of the TLS tunnel before the persistent timer expires, the server updates the newly established tunnel with DDT-related information.
4. If the TSC client fails to request restoration of the TLS tunnel before the persistent timer expires, the server deletes both the TLS tunnel and the associated DTLS tunnel.

The **show tscf statistics**, **show tscf tunnel all**, and **show tscf tunnel detailed** ACLI commands all provide various counts of HA tunnel restoration operations.

Nagle Algorithm Control

Like DDT, Nagle algorithm control can provide an alternative method of improving delivery of RTP media over a TCP/TLS transport.

The Nagle algorithm is defined in RFC 896, Congestion Control in IP/TCP Networks. The algorithm is designed to improve the efficiency of IP/TCP networks by reducing the number of transmitted packets. With the Nagle algorithm enabled, the transmitting device buffers any outgoing data until all previously sent data has been acknowledged, or until the size of the buffered data exceeds a full packet of output.

The Nagle algorithm is enabled by default, and is subject to user control.

Users can:

1. Specify a TSCF-Interface-specific default value (enable | disable) that determines the state of the Nagle algorithm on that TSCF interface

2. Override the default algorithm state on a specified tunnel

Override requests are included in the Application Payload as shown in TSC Client to TSC Server.

The **show tscf statistics**, **show tscf tunnel all**, **show tscf tunnel detailed**, and **show tscf tunnel** <tunnel-id> **verbose** ACLI commands all provide various counts of algorithm operations.

Refer to TSCF Interface Configuration for configuration details.

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is defined in RFC 2560, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. The protocol enables users to determine the revocation state of a specific certificate.

RFC 2560 specifies the data exchanged between an OCSP client, in the current implementation the TSCF, and an OCSP responder, the Certification Authority (CA), or its delegate that issued the certificate to be verified. An OCSP client issues a request to an OCSP responder and suspends acceptance of the certificate in question until the responder replies with a certificate status.

If the OCSP responder returns a status of **good**, the certificate is accepted and authentication succeeds. If the OCSP responder returns a status other than **good**, the certificate is rejected and authentication fails.

Certificate status is reported as:

- **Good**—Indicates a positive response to the status inquiry. At a minimum, a positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity period.
- **Revoked**—Indicates a negative response to the status inquiry. The certificate has been revoked, either permanently or temporarily.
- **Unknown**—Indicates a negative response to the status inquiry. The responder cannot identify the certificate.

When authentication of remote TSC clients is certificate-based, you can enable OCSP on TSCF ports to verify certificate status. The TSCF/OCSP responder connection must be secured by the TLS protocol.

OCSP Request Processing

With OCSP enabled, the TSCF checks certificate revocation during the TLS handshake. After receiving the client certificate, the TSCF sends an OCSP request to a responder, and pauses the TLS handshake. Assuming a good response, the TSCF resumes the handshake, and establishes a TLS tunnel once the handshake is successfully ended. If the response is **revoked** or **unknown**, the handshake fails, and TSCF does not establish a tunnel.

The TSCF places OCSP requests in an outbound queue, and maintains a list of outstanding requests, which is updated upon the receipt of OCSP responses. When the TSCF receives a new request, it checks the certificate serial number, and possibly the identity of the issuing CA, against the list. If the request matches a list entry, TSCF ignores the request, and awaits the response to the outstanding request; if the request does not match a list entry, TSCF sends a new OCSP request to a responder.

OCSP Certificate Verification

Providing OCSP services requires the creation of a secure TLS connection between a TSCF port and one or more OCSP responders.

This configuration is a three-step process:

1. Create one or more certificate-status-profiles. Each certificate-status-profile provides the information and cryptographic resources required to access a single OCSP responder.
2. Assign one or more certificate-status-profiles to a tls-profile. This tls-profile enables OCSP services and provides a list of one or more OCSP responders.
3. Assign the tls-profile to a TSCF port to enable OCSP service on that port.

For details on using the ACLI to configure OCSP-based certificate verification services, see TSCF OCSP Configuration.

TSC Server Configuration

TSC server configuration consists of the following steps; each step is identified as required or optional.

1. Configure TSCF global behavior (optional)

   Configuration of the *tscf-config* object specifies the handling of idle tunnel connections. Default values can be retained to enable standard tunnel processing.

2. Configure TSCF OCSP policy-based forwarding services (optional)

   The TSCF protocol policy configuration enables the forwarding of both tunneled and detunneled traffic depending on certain parameters in the packet headers. When packets are to be forwarded from one realm to another, or to a specific destination IP address as they are passing through the TSC server, create TSCF OCSP Protocol Policies. Packets may also be directed to a LDAP server for user authentication, or detunneled traffic may be sent to an XMPP server.

3. Configure one or more TSCF address pools (required)

   A tscf-address-pool specifies one or more contiguous ranges of IP addresses that are available for assignment to client applications. Later in the configuration process, you will assign an existing address pool to a TSCF interface. If the *tscf-address-pool* is used in conjunction with an optional *tscf-data-flow*, the combination provides a static egress route for tunneled traffic.

4. Configure one or more TSCF data flows (optional)

   This configuration while optional is almost always performed A *tscf-data-flow* operates in conjunction with a *tscf-address-pool* to provide a static egress route for tunneled traffic, usually to a gateway or application server.

5. Configure TLS or DTLS profiles (required for production environments)

   This configuration encrypts tunneled traffic using either TLS to encrypt streamed TCP packets or DTLS to encrypt UDP datagrams. In test/debug environments it can be advantageous to disable encryption. A *tls-profile* identifies the cryptographic resources available to ensure privacy using TLS or DTLS.

6. Configure one or more TSCF interfaces and ports (required)

   A *tscf-interface* and its associated *tscf-ports* provide server-side tunnel connectivity,

7. Enable DoS protection on TSCF ports (optional)

TSCF Global Configuration

TSCF global configuration specifies the handling of idle tunnel connections. By default, the TSC server transitions an idle tunnel client from the active to the persistent state after an idle period of 300 seconds. Assuming the tunnel remains idle for an additional 330 seconds, the TSC server then transitions the tunnel from the persistent to the closed state—tearing the tunnel down and releasing its resources. If this behavior is consistent with your deployment, no changes are required or encouraged at the TSCF global configuration level. If local network conditions require modification, adjust the **keepalive-timer**, **keepalive-timer-datagram**, and **tunnel-persistence-time** parameters as described below.

TSCF global configuration also enables and manages high-availability (HA) topologies — topologies in which a pair of SBCs operate in tandem, one in the active and the other in the backup role to provide reliable redundant operational availability. By default, HA is disabled. If your SBC operates in standalone mode (not part of an HA pair), you can safely ignore all HA parameters and simply retain default values. If operating as part of an HA pair, use the **red-port** parameter to enable HA as described in the following procedure. After enabling HA, Oracle Communications recommends the retention of default values for other HA parameters (**red-max-trans**, **red-sync-start-time**, and **red-sync-comp-time**) unless unusual local conditions require otherwise. Prior to modifying HA parameters, you should refer to the *ACLI Configuration Guide* for more detailed HA information.

Use the following procedure to perform TSCF global configuration.

1. From superuser mode, use the following command sequence to access tscf-config configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tscf
ORACLE(tscf)# tscf-config
ORACLE(tscf-config)#

keepalive-timer           keepalive interval (in seconds)
keepalive-timer-datagram  keepalive interval (in seconds) for tunnels
                          with datagram transport
tunnel-persistence-time   tunnel persistence time after disconnection
                          (should be greater than keep-alive time)
red-port                  tscf redundancy sync port: 0 to disable and 2004
                          to enable
red-max-trans             maximum redundancy transactions to keep on active
red-sync-start-time       timeout for transitioning from standby to active
red-sync-comp-time        sync request timeout after initial sync
                          compeletion
element-id                TSCF element Id, default : 0
options                   optional features/parameters
select                    select tscf config to edit
no                        delete tscf config
show                      show tscf config
done                      write tscf config information
exit                      return to previous menu

ORACLE(tscf-config)#
```

2. Use the **keepalive-timer**, **keepalive-timer-datagram**, and **tunnel-persistence-time** parameters to specify handling of idle tunnels.

The timing of SIK transmissions is defined by the TSCF global parameters, **keepalive-timer-datagram** and **keepalive-timer**. **keepalive-timer-datagram** is used for datagram-based (UDP) tunnels, while **keepalive-timer** is used for stream-based (TCP) tunnels. In the absence of a user-supplied value for **keepalive-timer-datagram**, **keepalive-timer** provides a default timer value for all tunnels regardless of the underlying transport protocol.

With SIK enabled, the server sends a keepalive message when no data is received from the client within the keepalive interval. The keepalive is sent approximately 15 seconds before timer expiration to ensure that it reaches the client before the client closes the tunnel for not receiving a keepalive message. For a datagram-based tunnel the TSC server waits 5 seconds for a keepalive response or other data from the client. If no response is received, the server sends another keepalive message. This sequence is repeated until the server receives a keepalive response or other tunnel data, or a total of 3 keepalive messages has been sent, whichever occurs first. If no keepalive response or other data is received from a client (either connection-based or datagram-based) within 15 seconds from the time the first keepalive message is sent, the server will tear down the tunnel, while saving the tunnel configuration if the if tunnel-persistence-timer registers a non-zero value.

keepalive-timer specifies the maximum idle time (defined as no transmission activity within the tunnel) before the TSC server transitions a stream-based (TCP) tunnel from the active to the persistent state. Supported values are **0** and integers within the range **30 - 550 (seconds)**, with a default value of 300. The special value **0** specifies that a keepalive mechanism is not employed. The integer values **30** through **550** specify the maximum size of the tolerated idle period for stream-based tunnels.

When **keepalive-timer-datagram** is set to its default value (0), keepalive-timer provides the default timer value for all tunnels, without regard to the transport protocol (TCP or UDP).

When **keepalive-timer** is set to a non-zero value, the TSC client must initiate a keepalive message exchange before an idle period exceeds the assigned non-zero value. Failure of the client to transmit a keepalive prior to the timer expiration results in the server placing the tunnel in the persistent state.

The complete keepalive message exchange consists of a client-generated Keep_Alive_Request and a server-generated Keep_Alive_Response—essentially empty control messages consisting only of address fields and a common message header.

The periodic exchange of keepalive messages not only verifies the presence of the remote tunnel peer, but also facilitates the maintenance of NAT bindings through strict firewalls.

keepalive-timer-datagram specifies the maximum idle time (defined as no transmission activity within the tunnel) before the TSC server transitions a datagram-based (UDP) tunnel from the active to the persistent state. Supported values are **0** and integers within the range **30 - 550 (seconds)**, with a default value of 0. The special value **0** specifies that a keepalive mechanism is not employed. The integer values **30** through **550** specify the maximum size of the tolerated idle period for datagram-based tunnels.

tunnel-persistence-time specifies the additional idle time tolerated before the TSC server transitions an idle tunnel from the persistent to the closed state and tears down the tunnel.

Allowed values are **0** and integers within the range **10 - 600** (seconds), with a default value of 330.

The value **0** specifies that the TSC server tears down the tunnel immediately upon expiration of the keep-alive timer. The integer values **10** through **600** specify the idle period, between transition from the persistent to the closed state.

During this interval, TSC clients, currently in the persistent state, can initiate a keepalive message exchange which serves to reset the keepalive timer and return the client to the

active state. During this same interval, recently disconnected clients, who are still in the persistent state, can re-connect and request their previous tunnel configuration.

For efficient and predictable operation, ensure that the value assigned to **tunnel-persistence-time** exceeds the value assigned to **keepalive-timer**.

```
ORACLE(tscf-config)# keepalive-timer 30
ORACLE(tscf-config)# tunnel-persistence-time 40
ORACLE(tscf-config)#
```

3. Use the **element-id** parameter to assign a unique identifier to this TSC server.

   Allowed values are 0 (the default) or an integer within the range **1** - **1023**.

   element-id can be safely ignored within network topologies that contain a single TSC server. However, in topologies that contain multiple TSC servers, each server must be assigned a unique network-wide identifier, provided by this parameter.

   The assignment of a unique TSC server guarantees unique tunnel identifiers regardless of the number of TSC servers within the network.

   ```
   ORACLE(tscf-config)# element-id 10
   ORACLE(tscf-config)#
   ```

4. In HA topologies use the **red-port**, **red-max-trans**, **red-sync-start-time**, and **red-sync-comp-time** parameters to configure reliable redundant operations.

   Use the **red-port** parameter to specify the UDP port number that supports TSCF synchronization message exchanges.

   The default value (**0**) disables redundant HA configurations. Use a port value of 2004 to enable HA operations.

   Use the **red-max-trans**parameter to specify the maximum number of retained TSCF synchronization messages.

   Allowed values are integers within the range 0 through 999999999 (messages) with a default of 10000.

   Use the **red-sync-start-time** parameter to set the timer value (in milliseconds) for transition from the standby to the active role — that is the maximum period of time that the standby SBC waits for a heartbeat signal from the active SBC before assuming the active role.

   Allowed values are integers within the range 0 through 999999999 (milliseconds) with a default of 5000.

   Use the **red-sync-comp-time** attribute to specify the interval between synchronization attempts after the completion of a TSCF redundancy check.

   Allowed values are integers within the range 0 through 999999999 (milliseconds) with a default of 1000.

   ```
   ORACLE(tscf-config)# red-port 2004
   ORACLE(tscf-config)# red-trans 15000
   ORACLE(tscf-config)# red-sync-start-time 2500
   ORACLE(tscf-config)# red-sync-comp-time 750
   ORACLE(tscf-config)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete TSCF global configuration.

TSCF Protocol Policy Configuration

Use the following procedure to configure TSCF policy-based forwarding services.

Policy-based forwarding requires the creation of a tscf-protocol-policy and the assignment of that policy to a tscf-address-pool.

1. From superuser mode, use the following command sequence to access cert-status-profile configuration mode. While in this mode, you configure a certificate-status-profile, a container for the information required to access a single, specific OCSP responder.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tscf
   ACMEPACKET(tscf)# tscf-protocol-policy
   ACMEPACKET(tscf-protocol-policy)#
   ```

2. The required **name** parameter differentiates tscf-protocol-policy instances from one another. Each tscf-protocol-policy specifies filtering/match criteria applied to incoming packets.

3. The optional **ip-address** parameter works in conjunction with the required **port** and **transport-type** parameters to identify incoming packets subject to this tscf-protocol-policy.

   For client-side, tunneled packets, **ip-address** is matched against the inner packet's destination **ip-address**. For non-tunneled packets, **ip-address** is matched against the source **IP-address**.
   For client-side, tunneled packets, **port** is matched against the inner packet's destination port. For non-tunneled packets, **port** is matched against the source port.

   Retain the default value (0.0.0.0) or specify a specific IP address.

4. The required **transport-type** parameter works in conjunction with the optional **ip-address** and required **port** parameters to identify incoming packets subject to this tscf-protocol-policy.

   For client-side, tunneled packets, **transport-type** is matched against the inner packet's transport protocol (UDP or TCP). For non-tunneled packets, **transport-type** is matched against the packet's transport protocol.

   Retain the default value (UDP), or select TCP.

5. For all tunneled packets that meet the matching criteria specified by the **ip-address**, **port**, and **transport-type** parameters, the required **realm-id** parameter specifies the egress realm. Matching packets are forwarded to the gateway that services the interface associated with the named realm.

6. Optionally, for all packets that meet the matching criteria specified by the **ip-address**, **port**, and **transport-type** parameters, the tunneled packets will be forwarded to the configured **remote-ip-address**. The destination IP address of the detunneled packet will be changed to this value.

   When the packets are coming from non-tunneled side to tunneled side, the source IP address must match the remote-ip-address. The source IP address will be changed back to the original destination IP address within the tunneled packet

7. Use **done**, **exit**, and **verify-config** to complete configuration of this tscf-protocol-policy.

8. Repeat Steps 1 through 7 to configure additional tscf-protocol-policies if necessary.

   For instructions on how to assign a TSCF protocol policy to a TSCF address pool, see the section TSCF Address Pool Configuration in this document. The parameter is called **protocol-policy**,and appears in Step 1 and Step 9.

TSCF Address Pool Configuration

During the configuration stage as described in TSCF Overview, the TSC server assigns a tunnel IP address to the client application. These assigned addresses are obtained by the TSC server from a tscf-address-pool, a configuration object that contains an IP address list. The IP address list contains one or more IP address ranges. Each address range consists of contiguous IP addresses, and can contain a minimum of 1, or a maximum of 262,144 list entries for IPv4.

The address range size, the list size, and the size of the tscf-address-pool are constrained by the same maximum value. Consequently, while the IP address list can contain one or several ranges, the total number of IP addresses contained in the individual address ranges cannot exceed 262,144 for IPv4.

Use the following required procedure to configure a tscf-address-pool configuration object. Later, you will assign the address pool to a specific TSCF interface.

1. From superuser mode, use the following command sequence to access tscf-address-pool configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tscf
ORACLE(tscf)# tscf-address-pool
ORACLE(tscf-address-pool)#

name            name for the tscf address pool
address-range   address ranges for the local address pool
dns-realm-id    dns realm identifier for local-pool
data-flow       data flow name for managing data traffic with the pool
protocol-policy comma separated protocol policy name/s for managing
                specific protocol traffic with the pool
batch           enter all arguments on one line
select          select a local address pool to edit
no              delete selected local address pool
show            show selected local address pool
done            write local address pool information
exit            return to previous menu

ORACLE(tscf-address-pool)#
```

2. Use the **name** parameter to provide a unique identifier for this tscf-address-pool.

```
ORACLE(tscf-address-pool)# name tscf-lap-1
ORACLE(tscf-address-pool)#
```

3. Retain the default value (an empty string) for the **dns-realm-id** parameter.
4. Use the **address-range** parameter to access address-range sub-configuration mode. While in this mode, you compile an IP address list, and assign contiguous ranges of IP addresses to the list.

```
ORACLE(tscf-address-pool)# address-range
ORACLE(tscf-address-range)# ?
network-address base network address for this range
subnet-mask     subnet mask for address range
batch           enter all arguments on one line
select          select a local address pool address range to edit
no              delete selected local address pool range
show            show selected local address pool range
done            write local address pool range information
```

```
exit            return to previous menu

ORACLE(tscf-address-range)#
```

5. Use the **network-address** and **subnet-mask** ACLI parameters to specify a contiguous range of IP addresses assigned to this tscf-address-pool.

   network-address can take any valid IP address value. **subnet-mask** values should take into consideration the projected number of tunnels to be supported by TSCF ports. To ensure that IPv4 address lists do not exceed the maximum supported value (262,144), mask values must begin with the initial 14 bits set to 1. The following table lists supported subnet-mask values along with the size of the resulting address pool.

   Supported **subnet-mask** Values Address Pool Size

255.252.0.0	262,144
255.254.0.0	131,072
255.255.0.0	65,536
255.255.128.0	32,768
255.255.192.0	16,384
255.255.224.0	8,192
255.255.240.0	4,096
255.255.248.0	2,048
255.255.252.0	1,024
255.255.254.0	512
255.255.255.0	256
255.255.255.128	128
255.255.255.192	64
255.255.255.224	32
255.255.255.240	16
255.255.255.248	8
255.255.255.252	4
255.255.255.254	2
255.255.255.255	1

   This command example allocates the maximum allowed address range to current the IPv4 address list.

```
ORACLE(tscf-address-pool)# network-address 10.244.0.0
ORACLE(tscf-address-pool)# subnet-mask 255.252.0.0
ORACLE(tscf-address-pool)#
```

   This command example allocates 4096 IP addresses to the IPv4 address list.

```
ORACLE(tscf-address-pool)# network-address 10.244.0.0
ORACLE(tscf-address-pool)# subnet-mask 255.255.240.0
ORACLE(tscf-address-pool)#
```

6. Use **done** and **exit** to add the address range to the IPv4 address list, and to return to **tscf-address-pool** configuration mode.

7. If necessary, repeat Steps 5 and 6 to add additional contiguous address ranges to the IPv4 address list.

   This command example allocates a second range of 16,384 IPv4 addresses, to the IPv4 address list.

```
ORACLE(tscf-address-pool)# network-address 172.16.240.0
ORACLE(tscf-address-pool)# subnet-mask 255.255.192.0
ORACLE(tscf-address-pool)#
```

8. Use the **data-flow** parameter to assign a specific **tscf-data-flow** to this **tscf-address-pool**.

Refer to TSCF Data Flow Configuration for configuration details.

```
ORACLE(tscf-address-pool)# data-flow tscf-df-1
ORACLE(tscf-address-pool)#
```

9. Use the **protocol-policy** parameter to assign specific protocol traffic to the pool. Use the name of the policy that was assigned to it when it was created. If assigning multiple policies, use a comma separated list.

   ```
   ACMEPACKET(tscf-address-pool)# protocol-policy <tscf-address-policy>
   ```

10. Use **done**, **exit**, **save**, and **verify-config** to complete configuration of the tscf-address-pool instance.

    verify-config confirms that a valid IP address and subnet mask are present in each IP address list, and that the total number of IPv4 addresses contained in the list does not exceed 261,144. If the verification process finds errors, **verify-config** issues appropriate explanatory error messages.

11. Repeat Steps 1 through 10 to configure additional **tscf-address-pool** instances.

TSCF Data Flow Configuration

Use the following procedure to configure an optional tscf-data-flow object. If you are not using tscf-data-flows to provide to provide static egress routes, this procedure can be safely ignored.

1. From superuser mode, use the following command sequence to access tscf-data-flow configuration mode. In this mode you configure tscf-data-flows that enable static pass-thru of tunneled data via a specified realm.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# tscf
   ORACLE(tscf)# tscf-data-flow
   ORACLE(tscf-data-flow)# ?
   name            name for a data flow
   realm-id        realm-id to route the upstream data flow
   group-size      the number of UEs to be managed by the data flow
   upstream-rate   Upstream bandwidth (KB/s) for the data flow
   downstream-rate Downstream bandwidth (KB/s) for the data flow
   batch           enter all arguments on one line
   select          select a data flow to edit
   no              delete selected data flow
   show            show selected data flow
   done            write data flow information
   exit            return to previous menu

   ORACLE(tscf-data-flow)#
   ```

2. Use the **name** parameter to provide a unique identifier for this tscf-data-flow instance.

   ```
   ORACLE(tscf-data-flow)# name tscf-dt-01
   ORACLE(tscf-data-flow)#
   ```

3. Use the **realm-id** parameter to identify the egress realm.

   ```
   ORACLE(tscf-data-flow)# realm-id core-1
   ORACLE(tscf-data-flow)#
   ```

4. Use the **group-size** parameter to specify the size and number of address blocks supported by this **data-flow** instance.

The size of the associated tscf-address-pool (defined as the number or IPv4 addresses contained in the pool), is divided by the **group-size** value to segment the address pool into smaller address blocks. After determining the start address for each block, the **SBC** establishes two static flows for each of the address blocks — a downstream flow, in the access direction, and an upstream flow generally toward a core application server gateway/ router that provides forwarding service for pass-thru data-flow.

As a result of the static-flow establishment, each address block consumes 4 NAT entries, 2 for the downstream flow and 2 for the upstream flow. Because the current software release imposes a maximum restriction of 4096 NAT entries per data-flow instance, dividing 4096 by 4 provides the largest number of supported address blocks, 1024.

Use the following table to associate supported **group-size** values with address pool size.

For example, for IPv4 addressing:

The largest supported address pool (262,144 addresses) requires a group size of 256 addresses, which produces the maximum 1,024 address blocks with each block containing 256 addresses.

An address pool containing 131,072 addresses can be segmented as 512 address blocks, with each block containing 256 addresses, or as 256 blocks, with each block containing the maximum 1,024 addresses.

An address pool containing 1,024 addresses can be segmented as follows:

1 block containing 1,024 addresses

2 blocks containing 512 addresses

4 blocks containing 256 addresses

8 blocks containing 128 addresses

16 blocks containing 64 addresses

32 blocks containing 32 addresses

64 blocks containing 16 addresses

128 blocks containing 8 addresses

256 blocks containing 4 addresses

Address Pool Size	Supported group-size Values
262,144	256
131,072	128, 256
65,536	64, 128, 256
32,768	32, 64, 128, 256
16,384	16, 32, 64, 128, 256
8,192	8, 16, 32, 64, 128, 256
4,096	4, 8, 16, 32, 64, 128, 256
2,048	2, 4, 8, 16, 32, 64, 128, 256
1,024	1, 2, 4, 8, 16, 32, 64, 128, 256
512	1, 2, 4, 8, 16, 32, 64, 128, 256
256	1, 2, 4, 8, 16, 32, 64, 128, 256
128	1, 2, 4, 8, 16, 32, 64, 128
64	1, 2, 4, 8, 16, 32, 64
32	1, 2, 4, 8, 16, 32
16	1, 2, 4, 8, 16
8	1, 2, 4, 8
4	1, 2, 4
2	1, 2
1	1

```
ORACLE(tscf-data-flow)# group-size 128
ORACLE(tscf-data-flow)#
```

5. Use the **upstream-rate** parameter to specify the allocated upstream (core-side) bandwidth.

   Allowable values are integers within the range **0** (the default) through **122070** (KB/s).

   The **0** value allocates all available upstream bandwidth.

   ```
   ORACLE(tscf-data-flow)# upstream-rate 100
   ORACLE(tscf-data-flow)#
   ```

6. Use the **downstream-rate** parameter to specify the allocated downstream (access-side) bandwidth.

   Allowable values are integers within the range **0** (the default) through **122070** (KB/s).

   The **0** value allocates all available downstream bandwidth.

   ```
   ORACLE(tscf-data-flow)# downstream-rate 100
   ORACLE(tscf-data-flow)#
   ```

7. Use **done**, **exit**, and **verify-config** to complete configuration of this tscf-data-flow instance.

   verify-config confirms that the total number of IPv4 addresses assigned to the data-flow instance does not exceed 262,144 and that the total number of NAT entries consumed by the data-flow instance does not exceed 4,096. In the event of discrepancies, **verify-config** issues an error message recording the actual number of IPv4 addresses and/or the number of NAT entries.

   Repeat Steps 1 through 7 to configure additional tscf-data-flow instances.

TLS Profile Configuration

Use the following required procedure to configure a tls-profile configuration object that identifies the cryptographic resources, specifically certificates and protocols, required for tunnel creation. Later, you will assign the tls-profile to a specific TSC port.

1. From superuser mode, use the following command sequence to access tls-profile configuration mode.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# security
   ORACLE(security)# tls-profile
   ORACLE(tls-profile)# ?

   name                      tls profile name
   end-entity-certificate    end entity certificate for the TLS connection
   trusted-ca-certificates   list of trusted certificate records
   cipher-list               list of ciphers
   verify-depth              maximum length of the certificate chain
   mutual-authenticate       mutually authenticate
   tls-version               TLS version
   options                   optional features/parameters
   cert-status-check         cert status check state
   cert-status-profile-list  list of cert-status-profiles for status requests
   ignore-dead-responder     Ignore dead cert status responder
   allow-self-signed-cert    allow self-signed certificate
   select                    select configuration to edit
   no                        delete configuration
   show                      show configuration information
   done                      write configuration information
   quit                      quit out of configuration mode
   ```

```
exit                          return to previous menu

ORACLE(tls-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this tls-profile.

```
ORACLE(tls-profile)# name tscfTls01
ORACLE(tls-profile)#
```

3. Use the required **end-entity-certificate** parameter to supply the unique name of a certificate-record configuration element referencing the identification credential (specifically, an X509.v3 certificate) offered by the TSC server in support of its asserted identity.

```
ORACLE(tls-profile)# end-entity-certificate tscfServerCert01
ORACLE(tls-profile)#
```

4. Use the required **trusted-ca-certificates** parameter to compile a list or one or more certificate-record configuration elements referencing trusted Certification Authority (CA) certificates used to authenticate a client application.

   Provide a comma separated list of existing CA **certificate-record** configuration elements.

```
ORACLE(tls-profile)# trusted-ca-certificates verisignClass3-a,verisignClass3-
b,baltimore,thawtePremium
ORACLE(tls-profile)#
```

5. Retain the default value, **all**, for the **cipher-list** parameter.

6. Use the **verify-depth** parameter to specify the maximum number of chained certificates that will be processed while authenticating the client application.

   Provide an integer within the range 1 through 10 (the default).

   The TSC server supports the processing of certificate chains when X.509v3 certificate-based authentication is used during tunnel establishment. The following process validates a received TLS certificate chain.

   a. Check the validity dates (Not Before and Not After fields) of the end certificate. If either date is invalid, authentication fails; otherwise, continue chain validation

   b. Check the maximum length of the certificate chain (specified by verify-depth). If the current chain exceeds this value, authentication fails; otherwise, continue chain validation.

   c. Verify that the Issuer field of the current certificate is identical to the Subject field of the next certificate in the chain. If values are not identical, authentication fails; otherwise, continue chain validation.

   d. Check the validity dates (Not Before and Not After fields) of the next certificate. If either date is invalid, authentication fails; otherwise, continue chain validation.

   e. Check the X509v3 Extensions field to verify that the current certificate identifies a CA. If not so, authentication fails; otherwise, continue chain validation.

   f. Extract the Public Key from the current CA certificate. Use it to decode the Signature field of the prior certificate in the chain. The decoded Signature field yields an MD5 hash value for the contents of that certificate (minus the Signature field).

   g. Compute the same MD5 hash. If the results are not identical, authentication fails; otherwise, continue chain validation.

   h. If the hashes are identical, determine if the CA identified by the current certificate is a trust anchor by referring to the trusted-ca-certificates attribute of the associated TLS-

profile configuration object. If the CA is trusted, authentication succeeds. If not, return to Step 2.

```
ORACLE(tls-profile)# verify-depth 8
ORACLE(tls-profile)#
```

7. Use the **mutual-authenticate** parameter to **enable** or **disable** (the default) mutual authentication.

   Protocol requirements mandate that the server present its certificate to the client application. Optionally, the server can implement mutual authentication by requesting a certificate from the client application, and authenticating the certificate offered by the client.

   Upon receiving a server certificate request, the client application must respond with a certificate; failure to do so results in authentication failure.

   ```
   ORACLE(tls-profile)# mutual-authenticate disabled
   ORACLE(tls-profile)#
   ```

8. Retain the default value, **compatibility**, for the **tls-version** parameter.
9. Retain default values for all other parameters.
10. Use **done**, **exit**, and **verify-config** to complete tls-profile configuration.
11. Repeat Steps 1 through 10 to configure additional tls-profiles as required.

TSCF OCSP Configuration

The following steps provide instruction on using the ACLI to configure OCSP-based certificate revocation services.

Providing OCSP services requires the creation of a secure TLS connection between a TSC port and one or more OCSP responders. This configuration is a three-step process.

1. Create one or more certificate-status-profiles. Each certificate-status-profile provides the information and cryptographic resources required to access a single OCSP responder.
2. Assign one or more certificate-status-profiles to a tls-profile. This tls-profile enables OCSP services and provides a list of one or more OCSP responders.
3. Assign the tls-profile to a TSCF port to enable OCSP service on that port.

Configure a certificate-status-profile

1. From superuser mode, use the following command sequence to access status-profile. cert-status-profile configuration mode. While in this mode, you configure a **certificate-status-profile**, a container for the information required to access a single, specific OCSP responder.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# cert-status-profile
   ACMEPACKET(cert-status-profile)#
   ```

2. The required **name** parameter differentiates certificate-status-profile instances from one another. Each certificate-status-profile provides configuration information for a single, specific OCSP responder.
3. The **type** parameter selects the certificate revocation check methodology; the only currently supported value is OCSP.

4. Retain the default value (http) for the **trans-protocol** parameter, which identifies the transport method used to access the OCSP responder.
5. The **ip-address** parameter works in conjunction with the **port** parameter to locate the OCSP responder.

   ip-address identifies the OCSP responder by its IP address. **port** identifies the port monitored by the responder for incoming OCSP requests.

   Allowable port values are integers within the range 1025 through 65535. In the absence of an explicitly configured value, a default value of 80, the well-known HTTP port, is provided.
6. The **realm-id** parameter specifies the realm used to transmit OCSP requests and receive OCSP responses. In the absence of an explicitly configured value, the default value of wancom0, specifying OCSP protocol transmissions across the wancom0 management interface, is used.
7. The **requester-cert** parameter is meaningful only if OCSP requests are signed; ignore this parameter if requests are not signed. RFC 2560 does not require the digital signature of OCSP requests. OCSP responders, however, can impose signature requirements.

   If a signed request is required by the OCSP responder, provide the name of the certificate configuration element that references the certificate used to sign OCSP requests.
8. The **responder-cert** parameter identifies the certificate used to validate signed OCSP response -- a public key of the OCSP responder. RFC 2560 requires that all OCSP responders digitally sign OCSP responses, and that OCSP requesters validate incoming signatures.

   Provide the name of the certificate configuration element that references the certificate used to validate the signed OCSP response.
9. The **retry-count** parameter specifies the maximum number of times to retry an OCSP responder in the event of connection failure. If the retry counter specified by this parameter is exceeded, the OCSP requester contacts another responder (if multiple responders have been configured) and quarantines the unavailable responder for a period defined by the dead-time parameter.

   In the absence of an explicitly configured value (an integer within the range 0 through 10), the default value of 1 (connection retries) is used.
10. The **dead-time** parameter specifies the quarantine period imposed on an unavailable OCSP responder. During the quarantine period, the TSC server will not send OCSP requests to the OCSP responder. In the absence of an explicitly configured value (an integer within the range 0 through 3600 seconds), the default value of 0 (no quarantine period) is used.

    > **Note:**
    >
    > Customer implementations utilizing a single OCSP responder are encouraged to retain the default value of the **dead-time** parameter, or to specify a brief quarantine period to prevent lengthy service outages.
11. The **hostname** and **trusted-ca** parameters can be safely ignored.
12. Use **done**, **exit**, and **verify-config** to complete configuration of this certificate-status-profile.
13. Repeat Steps 1 through 12 to configure additional certificate-status-profiles if necessary.

Assign the certificate-status-profile to a TLS profile

1. From superuser mode, use the following command sequence to access tls-profile configuration mode. While in this mode, you can alter an existing TLS profile to support OCSP transactions.
2. Use the **select** command to identify a specific tls-profile that will support OCSP requests and responses.
3. Set the **mutual-authenticate** parameter to enabled.
4. Use the **cert-status-check** parameter to enable certificate status checking on this tls-profile.

   Set this parameter to enabled.
5. Use the **cert-status-profile-list** parameter to assign an OCSP responder or responders to this TLS profile.

   For example, this ACLI sequence assigns a single OCSP responder.

   ```
   ACMEPACKET(tls-profile)#cert-status-profile-list "OCSP_Symantic OCSP_Thawte
   OCSP_Entrust"
   ACMEPACKET(tls-profile)#
   ```

   Each assigned certificate-status-profile contains the data needed to access a single OCSP responder. Responders are accessed in list order, from first to last. In the above example, the TSC server first contacts the Symantic OCSP responder. If the responder cannot be contacted within the limits specified by the retry-count parameter of the certificate-status-profile, the responder is quarantined for the time specified by the dead-time parameter, and the TSC server moves to the next list entry.
6. Use **done**, **exit**, and **verify-config** to complete this configuration.
7. If necessary, repeat this procedure to prepare other TLS profiles for OCSP-based certificate checking support.

Assign the tls-profile to a TSCF port

1. From superuser mode, use the following command sequence to access tscf-port configuration mode. While in this mode, you assign an existing TLS profile to a TSCF port.
2. Use the **select** command to identify a specific tscf port that will support OCSP requests and responses.
3. Use the **tls-profile** parameter to assign an OCSP-enabled tls-profile to the current TSCF port enabled.
4. Use **done**, **exit**, and **verify-config** to complete configuration.
5. If necessary, repeat this procedure to prepare other TSCF ports for OCSP-based certificate checking support.

Sample OCSP Configurations

certificate-status-profile configuration

A sample certificate-status-profile configuration follows:

```
ACMEPACKET# show running-config cert-status-profile
cert-status-profile
                     name                                    OCSP_Symantic
     ip-address                          192.0.2.100
     hostname
     port                                8080
     type                                OCSP
     trans-proto                         HTTP
     requestor-cert                      signOCSP
     responder-cert                      SymanticPublic-1
     trusted-cas
     realm-id                            wancom0
     retry-count                         1
     dead-time                           60
     last-modified-by                    admin@console
     last-modified-date                  2014-07-24 18:25:25
task done
ACMEPACKET#
```

This configuration creates a certificate-status-profile named OCSP_Symantic. The profile identifies an OCSP responder located at 192.0.2.100:8080. The required **responder-cert** parameter specifies the CA public certificate used by the TSC server to verify the signed OCSP response. The optional **requester-cert** parameter indicates that the OCSP responder requires signed requests, and identifies the certificate used by the TSC server to digitally sign OCSP requests. The optional **dead-time** parameter imposes a 60 second quarantine if the OCSP responder is unreachable. Retention of default values for the **realm-id** and **retry-count** parameters specify OCSP responder access via the wancom0 management interface and a retry count of 1.

tls-profile configuration

A sample tls-profile configuration follows:

```
ACMEPACKET# show running-config tls-profile
tls-profile
        name                                    TLS_OCSP
        end-entity-certificate                  TSCFCert_1
        trusted-ca-certificates                 CA_Symantic
                                                CA_Thawte
                                                CA_Entrust
                                                CA_DigiSign
        cipher-list                             All
        verify-depth                            10
        mutual-authenticate                     enabled
        tls-version                             compatibility
        cert-status-check                       enabled
                                cert-status-profile-
list                                                            OCSP_Symantic
                                                OCSP_Thawte
        ignore-dead-
responder
    disabled
        allow-self-signed-
cert
disabled
        last-modified-
by
                 admin@console
        last-modified-
date
```

```
          2014-07-24 19:40:37
task done
ACMEPACKET#
```

This configuration creates a tls-profile named TLS_OCSP. The profile uses the **mutual-authenticate** parameter to enable mutual authentication between the TSC server and the OCSP responders, the **cert-status-check** parameter to enable OCSP services, and the **cert-status-profile-list** parameter to identify three OCSP responders.

sample portion of a tscf-interface/tscf-port configuration

A sample portion of a tscf-interface/tscf-port configuration follows:

```
 ACMEPACKET# show running-config tscf-interface
tscf-interface
     realm-id                              access
     state                                 enabled
     max-tunnels                           200000
     local-address-pools                   pool1
           nagle-state                           enabled
     assigned-services                     SIP,redundancy,DDT,
                                           server-keepalive
                    tscf-port
           address                            172.16.21.2
        port                               443
        transport-protocol                 TLS
        tls-profile                        TLS_OCSP
...
...
...
        last-modified-by                     admin@console
        last-modified-date                   2014-07-24 19:51:03
task done
ACMEPACKET#
```

This configuration enables OCSP support on the TSCF port 172.16.21.2:443.

Monitoring OCSP Operations

The TSC server generates an SNMP trap when a configured OCSP responder becomes Operations unreachable. It generates second trap when connectivity is re-established with a previously unreachable OCSP responder.

The **show security ocsp stats** ACLI command provides OCSP operational counts.

- ```
  ACMEPACKET#show security ocsp stats
  18:49:34-147
  CERTD OCSP Statistics                        ---- Lifetime ----
                                              Recent        Total    PerMax
  Req Rcvd from Cavium                           0            0         0
  Resp Sent to Cavium                            0            0         0
  OCSP Req Sent                                  0            0         0
  OCSP Resp Rcvd                                 0            0         0
  OCSP Timeout                                      0            0         0
  Duplicate OCSP Request                            0            0         0
  OCSP Status-Good                                  0            0         0
  OCSP Status-Revoked                               0            0         0
  OCSP Status-Unknown                               0            0         0
  ```

  Key to OCSP Operational Counts:
  ```

- Req RCVD from Cavium—Number of certificates presented to the OCSP task for verification.
- Resp Sent to Cavium—Number of valid responses (good, revoked, unknown) received from OCSP responders.
- OCSP Req Sent—Number of requests sent to OCSP responders.
- OCSP RespRcvd—Number of responses received from OCSP responders.
- OCSP Timeout—Number of unanswered OCSP requests plus the number of internal timeouts.
- OCSP Status-Good—Number of good status responses received from OCSP responders.
- OCSP Status-Revoked—Number of revoked status responses received from OCSP responders.
- OCSP Status-Unknown—Number of unknown status responses received from OCSP responder.

TSCF Interface Configuration

TSCF interface configuration specifies the SBC IP address that is accessed by TSC clients to initiate tunnel creation, assigns resources that facilitate tunnel creation, identifies specific TSCF services offered by the interface, and limits the number of supported tunnels.

Use the following procedure to configure a TSCF interface—keeping in mind that

- a TSCF interface must be physically supported by an ETC NIU with a minimum of 8GB of installed DRAM
- a TSCF interface and SIP interface cannot coexist on the same network interface

1. From superuser mode, use the following command sequence to access tscf-interface configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tscf
ORACLE(tscf)# tscf-interface
ORACLE(tscf-interface)# ?

realm-id              realm identifier
state                 state
max-tunnels           maximum number of tunnels
local-address-pools   address pool name
assigned-services     comma-separated list of services
options               optional features/parameters
tscf-ports            list of TSCF ports
select                select tscf config to edit
no                    delete tscf config
show                  show tscf config
done                  write tscf config information
exit                  return to previous menu

ORACLE(tscf-interface)#
```

2. Use the optional **state** parameter to select the TSCF interface operational state.

   Supported values are **enabled** (the default) | **disabled**.

Retain the default value to place the TSCF interface in an active operational state. Use **disabled** to place the TSCF interface in a non-operational state.

```
ORACLE(tscf-interface)# state enabled
ORACLE(tscf-interface)#
```

3. Use the required **realm-id** parameter to identify the realm that supports TSC client access to the SBC.

   Enter the realm name.

   ```
   ORACLE(tscf-interface)# realm-id tsmAccess
   ORACLE(tscf-interface)#
   ```

4. Use the optional **nagle-state** parameter to select the operational mode of the Nagle algorithm on this TSCF interface.

   Retain the default value (enabled) to support Nagle algorithm operations on this interface; use the alternate value (disabled) to disable algorithm operations.

   ```
   ORACLE(tscf-interface)# nagle-state disabled
   ORACLE(tscf-interface)#
   ```

5. Use the optional **assigned-services** parameter to enable one of more services supported by this TSCF interface. Currently supported services are **ddt** (refer to Dynamic Datagram Tunneling), **nagle** (refer to The show tscf statistics, show tscf tunnel all, and show tscf tunnel detailed ACLI commands all provide various counts of HA tunnel restoration operations.), **redundancy** (refer to Tunnel Redundancy), **server-keepalive** (refer to Server-Initiated Keepalive), and **sip**.

   Retain the default value, **sip**, to enable tunneling of SIP traffic.

   Use the keyword, **ddt**, to enable the establishment of a linked tunnel pair—a persistent TCP/TLS tunnel to carry signalling data, and an on-demand UDP/DTLS tunnel for datagram traffic.

   Use the keyword, **nagle**, to support the override of the TSCF-interface-specific Nagle algorithm state (enabled | disabled) on an individual tunnel.

   If **nagle** is specific as an assigned service, the ACLI **verify-config** command issues a warning if it fails to find an existing TCP/TLS port.

   Use the keyword, **redundancy**, to enable tunnel redundancy for improved call quality under adverse packet loss.

   Use the keyword, **server-keepalive**, to enable the periodic generation of TSCF-initiated keepalive messages on an idle tunnel — defined as no transmission activity within the tunnel.

   To enable multiple services, for example SIP and tunnel redundancy, use **sip,redundancy**. The required comma (,) serves as an argument delimiter.

   ```
   ORACLE(tscf-interface)# assigned-services sip,redundancy
   ORACLE(tscf-interface)#
   ```

   To enable dynamic datagram tunneling, use **sip,ddt**.

   ```
   ORACLE(tscf-interface)# assigned-services sip,ddt
   ORACLE(tscf-interface)#
   ```

6. Use the optional **max-tunnels** parameter to specify the maximum number of concurrent tunnels supported by this TSCF interface.

This parameter allows available tunnel capacity (specified by the installed TSCF license) to be allocated across multiple realms. Allowable values are any integer from 0 (the default) to the maximum license capacity.

The special value, 0, indicates that the interface-specific limit is bounded only the licensed value.

```
ORACLE(tscf-interface)# max-tunnels 20000
ORACLE(tscf-interface)#
```

7. Use the required **local-address-pools** parameter to identify the local address pool that provides tunnel addresses to TSC clients.

   A specific address from the address pool will be transmitted to the TSC client as part of the tunnel creation process. This address provides the inner source address for tunneled packets originated by the TSC client application.

   Refer to TSCF Address Pool Configuration for address pool details.

   ```
   ORACLE(tscf-interface)# local-address-pools tscfPool172
   ORACLE(tscf-interface)#
   ```

8. Use the **tscf-port** parameter to move to tscf-port configuration mode.

   ```
   ORACLE(tscf-interface)# tscf-port
   ORACLE(tscf-port)# ?

   address              IP Address
   port                 port
   transport-protocol   transport protocol
   tls-profile          tls profile name
   select               select a tscf port to edit
   no                   delete selected tscf port
   show                 show tscf port information
   done                 write tscf port information
   exit                 return to previous menu

   ORACLE(tscf-port)#
   ```

9. Use the required **address** and **port** parameters to specify the IP address and port number monitored for incoming tunnel client messages.

   This address and port number will be transmitted to the TSC client as part of the tunnel creation process. This address provides the outer destination address for tunneled packets originated by the TSC client application.

   The TSCF port IP address must match the address range of the realm designated by the **realm-id** parameter. As previously stated, the physical interface must be provided by an ETC NIU.

   Enter a currently configured IP address and a port number with the range 1025 through 65535, with a default value of 5060.

   ```
   ORACLE(tscf-port)# address 192.168.11.22
   ORACLE(tscf-port)# port 443
   ORACLE(tscf-port)#
   ```

10. Use the optional **transport-protocol** parameter to identify the tunnel transport protocol.

    Supported values are **TLS** (the default, TCP/TLS) | **DTLS** (UDP/DTLS) | **TCP** | **UDP**.

> **Note:**
>
> The unsecured TCP and UDP protocols are only used for debug purposes, and are not intended for operational environments.

```
ORACLE(tscf-port)# transport-protocol TLS
ORACLE(tscf-port)#
```

11. If the **transport-protocol** parameter is either **TLS** or **DTLS**, use the required **tls-profile** parameter to identify the TLS profile that specifies the cryptographic resources available to support TLS or DTLS operations.

    If the specified tunnel transport protocol is either TCP or UDP (non-operational environments only), this parameter can be safely ignored.

    Refer to TLS Profile Configuration for details.

    ```
    ORACLE(tscf-port)# tls-profile tscfTLS01
    ORACLE(tscf-port)#
    ```

12. Use **done** and **exit** to return to tscf-interface configuration mode.
13. If required, repeat Steps 7 through 11 to configure additional TSCF ports.
14. Use **done**, **exit**, and **verify-config** to complete tscf-interface configuration.

    When validating the tscf-interface configuration **verify-config** confirms the following:

    - the presence of an ETC NIU equipped with a minimum of 8GB of installed DRAM, and issues an ERROR: MINOR ALARM message if a compliant NIU is absent
    - the realm specified by the realm-id parameter actually exists, and issues an error message in the case of a non-existent realm
    - the TSCF port address matches the address specified by the realm-id parameter
    - the presence of an associated SIP Interface, and issues an error message in the case of a non-existent realm
    - the TSCF address pool specified by the local-address-pool parameter actually exists, and issues an error message in the case of a non-existent address pool
    - the TSCF address pool is not mis-configured, and issues an error message if it is
    - the presence of one or more configured tscf-ports, and issues a warning message in the event of a missing port
    - if assigned-services is ddt,sip, the configuration of matching (same address and port values) stream-based and datagram-based tunnels — issues an error message in the case of a missing tunnel
    - if transport-protocol is TLS or DTLS, the assignment of a TLS profile to the port — issues an error message in the case of a no assignment or the assignment of a non-existent profile

    verify-config confirms that TSCF and SIP interfaces have not been configured on the same network interface. In the event of such a conflict, **verify-config** issues an ERROR message identifying the TSCF interface, the SIP interface, and the network interface by name. Additionally, **verify-config** places the TSCF interface in an inactive state, where it remains until the configuration has been repaired.

15. Repeat Steps 1 through 14 to configure additional *tscf-interfaces* as required.

TSCF DoS Protection Configuration

Use the following procedure to configure DoS protection as described in Denial of Service. DoS protection is assigned via the realm that supports the TSCF port.

1. From superuser mode, use the following command sequence to access realm configuration mode.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **select** command to identify the target realm, the realm supporting the TSCF port.

```
ORACLE(realm-config)# select
...
...
3. tsmAccess
...
...
ORACLE(realm-config)# 3
ORACLE(realm-config)#
```

3. Use the **access-control-trust-level** parameter to enable DoS protection.

   Set the parameter to **high** to enable DoS protection.

```
ORACLE(realm-config)# access-control-trust-level high
ORACLE(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete DoS configuration for the current realm.
5. Repeat Steps 1 through 4 to enable DoS on other TSCF ports.

TSCF Management Monitoring

A series of **show** commands provides data on:

- TSCF global statistics
- address pool usage
- tunnel creation and state

TSCF Global Statistics

The **show tscf statistics brief** command displays summary TSCF global data.

```
ORACLE# show tscf statistics brief
22:08:16-133
Tunnel Capacity=200000
                          -- Period --  ----- Lifetime ------
                    Active High   Total  Total   PerMax   High
Established Tunnels      0    0       0      0        0      0

22:08:16-133
TSCF Statistics       ----- Lifetime ------
                  Recent     Total   PerMax
Tunnel Adds            0         0        0
Tunnel Rejects         0         0        0
```

```
Tunnel Deletes         0         0        0
Tunnel Errors          0         0        0
ORACLE#
```

The **show tscf statistics** command displays verbose TSCF global data.

```
ORACLE# show tscf statistics
22:08:02-119
Tunnel Capacity=200000
                              -- Period --  ----- Lifetime ------
                        Active  High   Total  Total  PerMax    High
Established Tunnels          0     0       0      0       0       0

22:08:02-119
TSCF Statistics      ------ Lifetime -------
                     Recent  Total    PerMax
Tunnel Adds               0      0         0
Tunnel Rejects            0      0         0
Tunnel De1etes            0      0         0
Tunnel Errors             0      0         0

Active Tunnels                             : 1500
Established Tunnels                        : 1500
Finished Tunnels                           : 0
Released Tunnels                           : 1500
Max Active Tunnels                         : 0
Failed Tunnels - Malformed Request         : 0
Fai1ed Tunnels - Non Existing Tunnel Id    : 0
Fai1ed Tunnels - Out of Resources          : 0
Fai1ed Tunnels - Server Failure            : 0
Fai1ed Tunnels - Version Not Supported     : 0
ORACLE#
```

Address Pool Statistics

The **show tscf address-pool all** command displays address pool data for all configured address pools.

```
ORACLE# show tscf address-pool all

=========================================================================
#  |Addr Pool Name            |Total    |In Use   |Free     |Invalid  |
=========================================================================
1  |pool1                     |65279    |1500     |63779    |0        |
=========================================================================
ORACLE#
```

The **show tscf address-pool <addressPoolName>** command displays data for a single specified address pool.

```
ORACLE# show tscf address-pool pool1

=========================================================================
#  |Addr Pool Name            |Total    |In Use   |Free     |Invalid  |
=========================================================================
1  |pool1                     |65279    |1500     |63779    |0        |
=========================================================================
ORACLE#
```

Tunnel Statistics

The **show tscf tunnel all** command displays basic summary information for each tunnel.

```
ORACLE# show tscf tunnel all

=====================================================================
#      |    Tunnel Id    |    Inner IP    |    Outer IP    |St   |
=====================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77|  A  |
      2| 0000000100000043|      10.10.0.67|   192.168.50.77|  A  |
=====================================================================
ORACLE#
```

The **show tscf tunnel all detailed** command displays verbose information for each tunnel.

```
ORACLE# show tscf tunnel all detailed

Prot = Protocol
Att = Attributes. [6 Attributes]
      [R = Redundancy # = RedundacyFactor, P = Primary, D = DDT, S = Server
Initiated Keepalive/C= client Initiated Keepalive, T/F = Nagle On(T)/Off(F)
       H = Created on Standby]
Creation time and date (mm/dd/yy)
T1 = Time (seconds) since last data packet received from TSC
T2 = Time (seconds) since last data packet sent on TSC
T3 = Time (seconds) since last control message received from TSC
T4 = Time (seconds) since last control message sent on TSC
T5 = Time (seconds) in current state (Active or Persistent)
St = State [S=Start  No=NoSession  I=Init  Ne=Negotiating A=Active P=Persistent
R=Released]
RC = Reconnect count
=========================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=========================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
      2| 0000000100000043|      10.10.0.67|   192.168.50.77: 54458|...
=========================================================================
ORACLE#
```

The **show tscf tunnel peer-inner-ip <ipAddress>** command displays summary information for a specific tunnel identified by its IP address as assigned from an address pool.

```
ORACLE# show tscf tunnel peer-inner-ip 10.10.0.66

...

=========================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=========================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
=========================================================================
ORACLE#
```

The **show tscf tunnel peer-outer-ip <ipAddress>** command displays summary information for a set of tunnels emanating from the specified client interface address.

```
ORACLE# show tscf tunnel peer-outer-ip 192.168.50.77
```

```
...

=======================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=======================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
      2| 0000000100000043|      10.10.0.67|   192.168.50.77: 54458|...
=======================================================================
ORACLE#
```

The **show tscf tunnel state <state>** command displays basic tunnel data for all tunnels in the specified state.

<state> can take one of the following values: Established, Finished, Negotiating, NoSession, Released, Start, Unassigned.

```
ORACLE# show tscf tunnel state Active

...

=======================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=======================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
      2| 0000000100000043|      10.10.0.67|   192.168.50.77: 54458|...
=======================================================================
ORACLE#
```

The **show tscf tunnel tscf-interface <realmID>** command displays basic tunnel data for all tunnels supported by a TSCF port. The TSCF port is identified by the name of its supporting realm.

```
ORACLE# show tscf tunnel tscf-interface access

...

=======================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=======================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
      2| 0000000100000043|      10.10.0.67|   192.168.50.77: 54458|...
=======================================================================
ORACLE#
```

The **show tscf tunnel tunnel-id <tunnelID>** command displays basic tunnel data for a specific tunnel identified by its tunnel ID.

A tunnel ID is assigned by the TSC server during the configuration exchange, and can be most easily obtained with the **show tscf tunnel all** command.

```
ORACLE# show tscf tunnel tunnel-id 0000000100000042

...

=======================================================================
   #   |    Tunnel Id    |    Inner IP    |    Outer IP    : Port |...
=======================================================================
      1| 0000000100000042|      10.10.0.66|   192.168.50.77: 43354|...
=======================================================================
ORACLE#
```

Tunnel Deletion

ACLI CLI commands provide the ability to delete an individual tunnel, a group of tunnels, or all tunnels.

Use the **clear-tunnel tscf all** command to delete all tunnels.

```
ORACLE# clear-tunnel tscf all
WARNING: This will clear all tunnels and may cause service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 1502 TSCF tunnels.
ORACLE#
```

Use the **clear-tunnel tscf interface ...** command to delete a group of tunnels.

This command deletes all tunnels on a specified interface

```
ORACLE# clear-tunnel tscf M1:1
WARNING: This will clear all tunnels and may cause service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 753 TSCF tunnels.
ORACLE#
```

This command deletes all TLS tunnels on a specified TSCF port

```
ORACLE# clear-tunnel tscf M1:1 tls 192.168.11.22:5060
WARNING: This will clear all tunnels on this port and may cause
service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 753 TSCF tunnels.
ORACLE#
```

This command deletes all DTLS tunnels on a specified TSCF port

```
ORACLE# clear-tunnel tscf M1:1 dtls 192.168.11.22:5060
WARNING: This will clear all tunnels on this port and may cause
service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 226 TSCF tunnels.
ORACLE#
```

This command deletes all TCP tunnels on a specified TSCF port

```
ORACLE# clear-tunnel tscf M1:1 tcp 192.168.11.22:5060
WARNING: This will clear all tunnels on this port and may cause
service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 10 TSCF tunnels.
ORACLE#
```

This command deletes all UDP tunnels on a specified TSCF port

```
ORACLE# clear-tunnel tscf M1:1 udp 192.168.11.22:5060
WARNING: This will clear all tunnels on this port and may cause
service outage!!
Clear all TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 10 TSCF tunnels.
ORACLE#
```

Use the **clear-tunnel tscf tunnel-id ...** command to delete a single tunnel.

```
ORACLE# clear-tunnel tscf tunnel-id feed3eef000000cf
WARNING: This will clear this tunnel and may cause service outage!!
Clear this TSCF tunnels ? [y/n]?: y
Command was successful. Deleted 1 TSCF tunnels.
ORACLE#
```

22
RCS Services

Message Session Relay Protocol

The Oracle USM supports Message Relay Protocol (MSRP) sessions initiated by Session Description Protocol (SDP) messages exchanged via the Session Initiation Protocol (SIP) offer/answer model. MSRP usage with SDP and SIP is described in Section 8 of RFC 4975, The Message Relay Protocol. The Oracle USM functions as a Back-to-Back User Agent (B2BUA) for MSRP sessions, terminating incoming MSRP, proxying for the MSRP session originator, initiating outgoing MSRP to the endpoint peer, and providing Network Address Translation (NAT) services.

MSRP Platform Requirements

MSRP is supported on the following platforms:

- Acme Packet 4500
- Acme Packet 4600
- Acme Packet 6100
- Acme Packet 6300

The Acme Packet 3820 does not support MSRP.

Users with Acme Packet 4500 systems must have an ETCv1 or ETCv2 NIU card installed in their systems. The Acme Packet 4600, Acme Packet 6100 and Acme Packet 6300 have the requisite hardware in their systems by default.

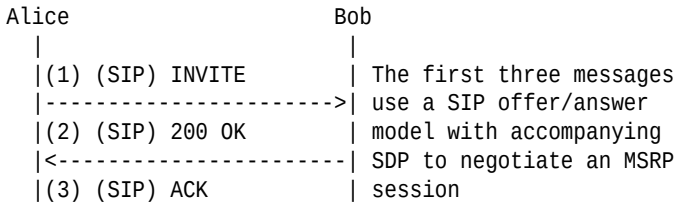
The Oracle USM now supports MSRP over IPv6 and IPv4/IPv6 interworking for MSRP.

MSRP IP Address Family Support

The Oracle USM supports MSRP over IPv4 and IPv6. The Oracle USM also can perform IPv4-to-IPv6 and IPv6-to-IPv4 interworking. This support is available automatically and does not require any configuration.

MSRP Operational Description

A sample RFC 4975-compliant Offer/Answer SDP exchange for an MSRP session is shown below.



```
|---------------------->|
|(4) (MSRP) SEND        | Message 4 starts the MSRP
|---------------------->| connection
|(5) (MSRP) 200 OK      |
|<----------------------|
|(6) (SIP) BYE          |
|---------------------->| Message 7 terminates the
|(7) (SIP) 200 OK       | SIP and MSRP connection
|<----------------------|
```

1. Alice sends an INVITE request with accompanying SDP to Bob.

   The SDP media (M) line is defined in RFC 4975, and adheres to the format

   m=<media> <port> <protocol> <format-list>

   MSRP operations require the following values:

media	=	message
protocol	=	TCP/MSRP (for an unencrypted connection)
format-list	=	*
port	=	the TCP port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for a response to the SDP offer

   The required SDP attributes, accept-types and path, are also defined in RFC 4975.

   accept-types contains a list of media types that the message originator is willing to receive. It may contain zero or more registered media-types, or an * wildcard character in a space-delimited string.

   path contains the MSRP URI of the message originator. An MSRP URI is constructed as shown below.

scheme	=	msrp (for an unencrypted connection), or
	=	msrps (for an encrypted connection)
//		
address	=	IP address of the message originator, or
	=	FQDN of the message originator
;		
port	=	the port (7777 in the SDP example, although any valid port number can be specified) monitored by the message originator for MSRP responses
session-id	=	a random local value generated by the message originator used to produce an ephemeral MSRP URI lasting only for the duration of the current MSRP session
;		
protocol	=	tcp

```
Alice->Bob (SIP):
```

```
INVITE sip:bob@example.com
SDP:
v=0
o=alice 2890844557 2890844559 IN IP4 alicepc.example.com
s=
c=IN IP4 alicepc.example.com
m=message 7777 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://alicepc.example.com:7777/iau39soe2843z;tcp
```

2. Bob accepts the SDP offer, generates a local session-id (contained in his MSRP URI specified by the path attribute), and issues a 200 OK response to Alice.

   The port parameter in the Media line indicates that Bob listens for MSRP messages on TCP port 8888

```
Bob->Alice (SIP):

SIP/2.0 200 OK
SDP:
v=0
o=bob 2890844612 2890844616 IN IP4 bob.example.com
s=
c=IN IP4 bob.example.com
m=message 8888 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://bob.example.com:8888/9di4eae923wzd;tcp
```

3. Alice ACKs Bob's answer, establishing a SIP session between the two.

```
Alice->Bob (SIP):

ACK sip:bob@example.com
```

4. Alice initiates an MSRP session with an MSRP SEND request to Bob.

   All MSRP requests begin with the MSRP start line, which contains three elements.

protocol-id	=	MSRP
transaction-id	=	an ephemeral transaction identifier (d93kswow in the following MSRP example) used to correlate MSRP requests and responses, and to frame the contents of the MSRP message
method	=	SEND (MSRP method that supports data transfer)

The MSRP start line is followed by the To-Path and From-Path headers, which contain destination and source addresses — the MSRP URIs exchanged during the MSRP negotiation.

The Message-ID header contains a random string generated by the message originator. This ephemeral value whose lifetime is measured from message start to message end, is used to correlate MSRP status reports with a specific message, and to re-assemble MSRP message fragments (chunks in MSRP terminology).

The Byte-Range header contains the message length, in bytes, and the specific byte range carried by this message. Contents of this header are generally of interest only if the message has been fragmented.

The Content-Type header describes the message type, and must conform to the results of the MSRP negotiation.

The actual message follows the Content-Type header.

Finally, the SEND request is closed with an end-line of seven hyphens, the transaction-id, and a

$ to indicate that this request contains the end of a complete message, or

+ to indicate that this request does not contain the message end

```
Alice->Bob (MSRP):

MSRP d93kswow SEND
To-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
From-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
Message-ID: 12339sdqwer
Byte-Range: 1-16/16
Content-Type: text/plain
Hi, I'm Alice!
-------d93kswow$
```

5. Bob acknowledges receipt with an MSRP 200 OK response to Alice.

   Note that the response includes the initiator-originated transaction-id, d93kswow.

   ```
   Bob->Alice (MSRP):

   MSRP d93kswow 200 OK
   To-Path: msrp://alicepc.example.com:7777/iau39soe2843z;tcp
   From-Path: msrp://bob.example.com:8888/9di4eae923wzd;tcp
   -------d93kswow$
   ```

6. Alice sends a BYE request to Bob.

   Alice sends a BYE request to terminate the SIP session and MSRP sessions. Alice can, of course, send more SEND requests to Bob before sending the BYE.

   ```
   Alice->Bob (SIP):

   BYE sip:bob@example.com
   ```

7. Bob sends a 200 OK response to Alice.

   ```
   Bob->Alice (SIP):

   SIP/2.0 200 OK
   ```

   The SIP session and the MSRP session are terminated.

Secure MSRP Session Negotiation

An Offer/Answer SDP exchange for a TLS (secure) MSRP session is specified in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP). RFC 4572 defines the syntax and symantics for an SDP fingerprint attribute that identifies the certificate (most likely a self-signed certificate) that will be presented during the TLS negotiation. A sample SDP exchange is shown below.

The protocol field (TCP/TLS/MSRP) of the media (M) line designates a TLS encrypted connection.

The fingerprint attribute is constructed as follows:

protocol	identifies the hashing method used to produce the certificate fingerprint, SHA-1 in the following sample SDP

hash value	a sequence of uppercase hexadecimal bytes separated by colons with the sequence length determined by the hash function

Offer SDP: (Alice to Bob)

```
v=0
o=usera 2890844526 2890844527 IN IP4 alice.example.com
s=
c=IN IP4 1.1.1.1
m=message 7394 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://alice.example.com:7394/2s93i9ek2a;tcp
a=fingerprint:SHA-1
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Answer SDP: (Bob to Alice)

```
v=0
o=userb 2890844530 2890844532 IN IP4 bob.example.com
s= -
c=IN IP4 2.2.2.2
t=0 0
m=message 8493 TCP/TLS/MSRP *
a=accept-types:text/plain
a=path:msrps://bob.example.com:8493/si438dsaodes;tcp
a=fingerprint:SHA-1
DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:AF:D8:07:09
```

MSRP Session Setup

The B2BUA processes MSRP media descriptions in offer/answer SDPs to negotiate and establish MSRP sessions and then constructs internal data flows for the actual MSRP sessions. After establishing an MSRP session, the B2BUA forwards MSRP requests and responses from and to the session participants.

Initiating MSRP Sessions

After accepting an offer SDP with MSRP session initiation, the B2BUA constructs an egress offer SDP as follows.

1. The B2BUA sets the transport protocol of the m= line to the transport protocol of the selected egress profile.
2. If the value of listen-port of the selected egress profile is not zero, the B2BUA sets the port of the m= line to the value of listen-port. If the value of listen-port is zero, the port in the m= line is chosen from a steering port of the egress realm.
3. The B2BUA adds an a=setup attribute to the SDP. The value of this attribute is determined by the value of the **preferred-setup-role** ACLI command. For example, if the value of **preferred-setup-role** is passive (the default value) the B2BUA adds the attribute a=setup:passive.
4. The B2BUA performs NAT on the MSRP Universal Resource Identifier (URI) in the a=path attribute.

   The B2BUA does not include an a=fingerprint in the offer SDP if the selected egress profile has TCP/TLS/MSRP transport protocol. However, if the egress profile specifies both TCP/MSRP and TCP/TLS/MSRP the B2BUA selects the TCP/TLS/MSRP transport protocol, resulting in an egress offer containing an m= line with TCP/TLS/MSRP transport

protocol. The B2BUA offers only a single media line, TCP/MSRP or TCP/TLS/MSRP. The B2BUA does not perform recursion (that is, first initiation attempt with TCP/TLS/MSRP and re-attempt with TCP/MSRP if first attempt is rejected).

Connection Negotiation

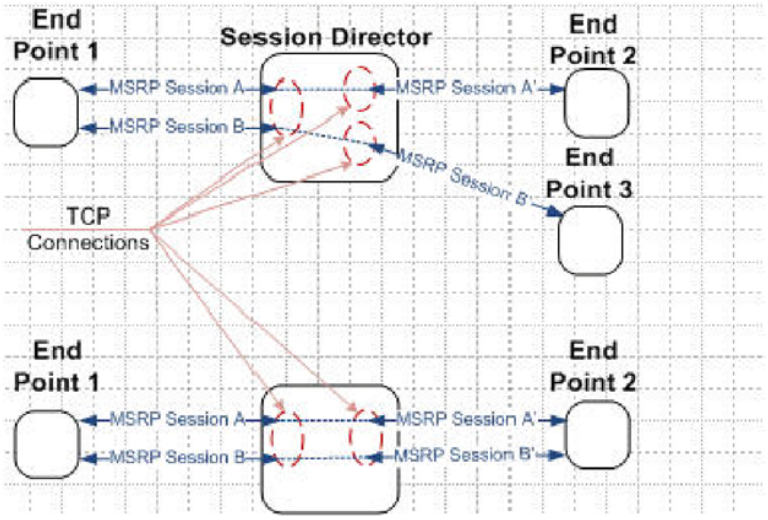
When making the offer MSRP session, a user agent client can follow RFC 4145, TCP-Based Media Transport in the Session Description Protocol (SDP), and include an a=setup attribute in a MSRP media line, or it may assume the default connection direction as specified in section 8.5 of RFC 4975, The Message Session Relay Protocol, which specifies that the endpoint that sent the original offer is responsible for connecting to its remote peer. The B2BUA, in contrast, always includes an a=setup attribute in its offer SDP, the attribute value set by the **preferred-setup-role** ACLI command.

If the B2BUA receives an answer SDP that does not include an a=setup attribute, it assumes that the user agent server does not support connection negotiation per RFC 4145. In that case, the B2BUA assumes the active role as specified in RFC 4975, and makes an outgoing connection.

If the B2BUA receives an offer or answer SDP that does include an a=setup attribute, it follows the connection negotiation as specified by RFC 4145. When the B2BUA receives an offer SDP with the attribute a=setup:actpass attribute, it will set the a=setup attribute of the answer SDP to the value set by the **preferred-setup-role** ACLI command.

Multiple MSRP Connections

The MSRP B2BUA supports sharing of a single TCP connection by multiple MSRP sessions. In such a topology, each TCP connection maintains a list active MSRP sessions.



With regard to the above figure, the upper Oracle USM's TCP connection between Endpoint 1 and the MSRS B2BUA is shared by 2 MSRP sessions. At bottom, each MSRP session uses a separate TCP connection. When the B2BUA assumes the active role, it always initiate a separate connection to the MSRP endpoint peers, endpoints 2 and 3 as shown above.

When the list of active MSRP sessions for a shared TCP connection becomes empty as a result of SIP session terminations or disconnections of peer TCP connections, B2BUA disconnects the shared TCP connection. If the shared connection is disconnected by the peer, the B2BUA disconnects all the separate TCP connections it initiated for the MSRP sessions that use the shared connection.

Accepting Connections

When the B2BUA is in passive mode, it listens for incoming connection from the active party, monitoring the port specified by the **listen-port** ACLI command.

Making Connections

When the B2BUA is in active mode, it makes the connection to the passive party, selecting a port allocated from the steering-pool of the applicable realm.

MSRP Session Termination

An MSRP session is terminated by sending or receiving a BYE request in the parent SIP session, that is the session that set up the MSRP exchange, followed by the disconnect of the TCP connection that supports MSRP message exchange.

Some SIP endpoints close their MSRP TCP connections upon receiving a BYE possibly before its peer finishes sending all the MSRP messages and closes the connection. To facilitate graceful session completion, the B2BUA offers a configurable time delay between the receipt of a BYE request from an MSRP endpoint and the transmission of the BYE to the recipient endpoint peer.

With the configurable BYE delay enabled, the MSRP B2BUA upon receiving a BYE request acknowledges the request with a 200 OK response. The B2BUA, however, does not immediately forward the BYE to the other MSRP endpoint.

Rather, the B2BUA triggers two user-configurable timers that monitor the specific MSRP session initiated by the current SIP exchange. The first timer measures inactivity intervals on media flows (calling-to-called and called-to-calling) associated with the MSRP session to be closed. The second timer sets an unconditional outer limit at which point the delayed BYE is transmitted to the MSRP endpoint and the MSRP is terminated.

Expiration of either timer generates an internal stop event which generates transmission of the delayed BYE to the MSRP endpoint and tem ina ti on of the underlying SIP connection.

Note that the MSRP-specific timers, the session inactivity timer and the MSRP delayed-BYE timer, are roughly analogous to two existing TCP timers, the TCP subsequent guard timer and the TCP flow time limit timer. The following sections summarize timer operations.

MSRP interval timer

- Purpose: Measures inactivity periods within MSRP data sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.
- Associated ACLI Command: **session-inactivity-timer**
- Allowable command values: 0 | 5 to 10 (seconds). When set to 0, session monitoring is disabled. No MSRP session monitoring is done when the corresponding SIP session receives a BYE request.
- Default value: 5
- Timer Expiration: Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP subsequent guard timer

- Purpose: Measures the maximum interval allowed between media-over-TCP packets.
- Associated ACLI Command: **tcp-subsq-guard-timer**
- Allowable command values: 0 to 999999999 (seconds)
- Default value: 300
- Timer Expiration: Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP interval timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

MSRP delayed -BYE timer

- Purpose: Measures inactivity periods within MSRP traffic sessions. The timer is triggered in the absence of MSRP data. If new MSRP is not detected prior to timer expiration, a stop event is generated resulting in delayed BYE transmission and MSRP connection termination. If new MSRP traffic is detected prior to timer expiration, the timer is reset.
- Associated ACLI Command: **msrp-delayed-bye-timer**
- Allowable command values: 0 to 60 (seconds)
- Default value: 15
- Timer Expiration: Initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the SIP connection.

TCP flow time limit timer

- Purpose: Measures the maximum allowed lifetime of a media-over-TCP connection.
- Associated ACLI Command: **tcp-flow-limit-timer**
- Allowable command values: 0 to 999999999 (seconds)
- Default value: 86400 (1 day)
- Timer Expiration: Initiates tear down of the TCP connection. In the possible, but unlikely event that the value assigned to this timer is less than the value assigned to the MSRP delayed-BYE timer, expiration initiates forwarding of the delayed BYE request to the recipient MSRP endpoint and tear down of the TCP connection.

Network Address Translation

If the value of the **uri-translation** ACLI command is **enabled**, the B2BUA performs network address translation on the MSRP URI in the a=path attribute and in the From-Path and To-Path of MSRP requests and response. The host part of the URI will be the IP address of the steering pool of the realm. The port will be chosen as follows:

If the B2BUA role is in passive mode, and the listen-port ACLI command is non-zero, the B2BUA monitors that specified port.

If the B2BUA role is in passive mode, and the listen-port ACLI command is zero, the B2BUA selects a port from the steering pool of the applicable realm and monitors that chosen port.

If the B2BUA role is in active mode, the port is chosen from the steering pool of the applicable realm. Since B2BUA is active, that port is used only to support the outgoing connection.

Certificate Fingerprint

If the value of the field **require-fingerprint** in the ingress and/or egress tcp-media-profile is enabled, and the transport protocol is TCP/TLS/MSRP the B2BUA requires the offer and/or the answer SDPs, respectively, to have an a=fingerprint attribute as specified in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP).

If the B2BUA receives an offer SDP without a=fingerprint attribute, it rejects the offer SDP. If the rejected SDP is in an INVITE request, the B2BUA issues a 488 Not Acceptable Here response. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a 503 Service Unavailable response to the client user agent.

If the B2BUA receives an answer SDP without a a=fingerprint attribute, it terminates the related SIP session. If the rejected SDP is in a 200 OK response, the B2BUA ACKs that 200 OK, sends a BYE to the server user agent, and a 503 Service Unavailable response to the client user agent. If the rejected SDP is in an ACK, the B2BUA simply sends a BYE to both the server and client user agent.

If the value of the field **require-fingerprint** in the profile is disabled, the B2BUA accepts offer and answer SDP that do not include an a=fingerprint attribute.

Because the B2BUA certificate is expected to be signed by a trusted Certification Authority, an a=fingerprint attribute is not included in offer and answer SDPs sent by the B2BUA.

The B2BUA maintains a fingerprint mismatch counter for each MSRP session. This counter is incremented when the B2BUA cannot match the certificate it receives during the TLS handshake with the fingerprint it receives in the SDP.

MSRP Configuration

MSRP configuration consists of the following steps.

1. Configure the msrp-config configuration object that governs MSRP global behavior.
2. Configure one or more tcp-media-profile configuration objects that define MSRP operations within a realm.
3. Assign a tcp-media-profile to a target realm.
4. If MSRP sessions are secured with TLS, create and assign tls-profile configuration objects to the tcp-media-profile of the target realm.
5. Create and assign steering-pools configuration objects to target realms.

msrp-config Configuration

Use the following procedure to perform MSRP global configuration.

1. From superuser mode, use the following command sequence to access msrp-config configuration mode. While in msrp-config mode, you configure global MSRP behavior.

```
ORACLE# configure terminal
ORACLE(configure)# media-manager
ORACLE(media-manager)# msrp-config
ORACLE(msrp-config)# ?
state                    state
```

```
uri-translation          perform translation of MSRP URI
session-inactivity-timer timer value (seconds) for session inactivity
                         monitoring period
select                   select msrp config to edit
no                       delete msrp config
show                     show msrp config
done                     write msrp config information
exit                     return to previous menu
ORACLE(msrp-config)#
```

2. Use the **state** parameter to enable MSRP operations.

   Retain the default value, **enabled**, to enable MSRP operations.

   If necessary, you can use **disabled** to temporarily suspend all MSRP operations.

   ```
   ORACLE(msrp-config)# state enabled
   ORACLE(msrp-config)#
   ```

3. Use the **uri-translation** parameter to enable or disable NAT of URIs found in the From-Path and To-Path headers of MSRP requests and responses, and in a=path attributes found in SDP offers.

   NAT is enabled by default.

   Retain the default value (**enabled**) to enable NAT; use **disabled** to disable NAT.

   ```
   ORACLE(msrp-config)# uri-translation enabled
   ORACLE(msrp-config)#
   ```

4. Use the **session-inactivity-timer** parameter in connection with the **msrp-delayed-bye-timer** parameter to implement the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

   The **session-inactivity-timer** parameter specifies the maximum inactivity interval (defined as the absence of transmitted data) tolerated before the MSRP connection is terminated.

   Retain the default value (**5**), or specify another inactivity interval within the range 5 to 10 seconds.

   ```
   ORACLE(msrp-config)# session-inactivity-timer 7
   ORACLE(msrp-config)#
   ```

5. Use **done**, **exit**, and **verify-config** to complete MSRP global configuration.

6. If you wish to implement the delayed transmission of SIP BYE requests, use the following command sequence to access sip-config configuration model

   ```
   ORACLE# configure terminal
   ORACLE(configure)# session-router
   ORACLE(session-router)# sip-config
   ORACLE(sip-config)#
   ```

7. Use the **msrp-delayed-bye-timer** parameter to enable the delayed transmission of SIP BYE requests, thus establishing a configurable transition interval allowing for the completion of active MSRP sessions.

   The **msrp-delayed-bye-timer** parameter specifies the maximum delay period allowed before transmitting the delayed BYE request.

   Retain the default value (1**5**), or specify another delay period within the range 1 to 60 seconds.

   Delayed transmission of BYE requests is enabled by default. Use the special value of 0 to disable delay, and transmit BYE requests immediately upon receipt.

```
ORACLE(sip-config)# msrp-delayed-bye-timer 20
ORACLE(sip-config)#
```

tcp-media-profile Configuration

Use the following procedure to construct a TCP Media Profile that defines MSRP operations within a realm.

1. From superuser mode, use the following command sequence to access tcp-media-profile configuration mode. While in tcp-media-profile mode, you begin construction of a TCP Media Profile.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# tcp-media-profile
   ORACLE(tcp-media-profile)# ?
   name          name
   profile-list  list of TCP media profiles
   select        select profile to edit
   no            delete profile
   show          show profile
   done          write profile information
   exit          return to previous menu
   ORACLE(tcp-media-profile)#
   ```

2. Use the **name** parameter to provide a unique identifier for this TCP Media Profile instance.

   ```
   ORACLE(tcp-media-profile)# name tlsMSRP
   ORACLE(tcp-media-profile)#
   ```

3. Use the **profile-list** command to move to tcp-media-profile-entry configuration mode. While in this mode, you complete configuration of the named tls-media-profile.

   ```
   ORACLE(tcp-media-profile)# profile-list
   ORACLE(tcp-media-profile-entry)# ?
   media-type            media type
   transport-protocol    transport protocol
   listen-port           listening port
   preferred-setup-role  preferred setup role
   tls-profile           tls profile name
   require-fingerprint   always require TLS certificate fingerprint
   select                select a profile entry to edit
   no                    delete selected profile entry
   show                  show profile entry information
   done                  write profile entry information
   exit                  return to previous menu
   ORACLE(tcp-media-profile-entry)#
   ```

4. Use the **listen-port** parameter to identify the TCP port monitored by the B2BUA for incoming MSRP connections.

   Supported values are integer values within the range 0 (the default value) through 65535.

   The 0 default value indicates that the listening port will be chosen by the B2BUA from the steering pool of the realm (which the tcp-media-profile belongs to).

   ```
   ORACLE(tcp-media-profile-entry)# listen-port 43000
   ORACLE(tcp-media-profile-entry)#
   ```

5. Use the **media-type** parameter in conjunction with the **transport-protocol** parameter to identify the media-types and transport protocols (found in the SDP media description, m=, field as described in RFC 4566, SDP: Session Description Protocol) subject to this TCP Media Profile.

media-type identifies the media subject to this TCP Media Profile. Retain the default value, **message**, for MSRP operations.

transport-protocol identifies the transport layer protocols subject to this TCP Media Profile. Use either **TCP/MSRP** to specify unsecured TCP traffic or **TCP/TLS/MSRP** to specify secured/encrypted TLS traffic.

```
ORACLE(tcp-media-profile-entry)# transport-protocol TCP/TLS/MSRP
ORACLE(tcp-media-profile-entry)#
```

6. If **transport-protocol** is **TCP/TLS/MSRP**, use the **tls-profile** parameter to identify the TLS profile that specifies the cryptographic resources available to support TLS operations.

   This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

   ```
   ORACLE(tcp-media-profile-entry)# tls-profile msrp1
   ORACLE(tcp-media-profile-entry)#
   ```

7. If **transport-protocol** is **TCP/TLS/MSRP**, use the **require-fingerprint** parameter to enable or disable endpoint authentication using the certificate fingerprint methodology defined in RFC 4572, Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP).

   By default, mutual authentication is **disabled**.

   This parameter can be safely ignored if **transport-protocol** is **TCP/MSRP**.

   ```
   ORACLE(tcp-media-profile-entry)# require-fingerprint enabled
   ORACLE(tcp-media-profile-entry)#
   ```

8. Use the **preferred-setup-role** parameter to specify the value the B2BUA uses for the a=setup attribute when negotiating the setup up role, regardless of the role (offerer or answerer) assumed by the B2BUA in the SDP offer/answer exchange.

   The value of **preferred-setup-role** is used for the value of the a=setup attribute when the B2BUA makes an offer SDP and when the B2BUA replies to an offer SDP that has a=setup:actpass. It is not used when the B2BUA is forced into a role by the offerer, that is, if the offerer sends a=setup:active, the B2BUA must answer with a=setup:passive (and vice versa).

   Allowable values are passive (the default) and active.

   active indicates that the B2BUA creates an outgoing connection.

   passive indicates that the B2BUA accepts an incoming connection.

   Acme Packet strongly recommends that users retain the default value, passive.

   ```
   ORACLE(tcp-media-profile-entry)# preferred-setup-role passive
   ORACLE(tcp-media-profile-entry)#
   ```

9. Use **done**, **exit**, and **verify-config** to complete tcp-media-profile configuration.
10. Repeat Steps 1 through 9 to configure additional tcp-media-profiles as required.

realm Configuration

Use the following procedure to assign a single, specific tcp-media-profile to a target realm.

1. From superuser mode, use the following command sequence to access realm-config configuration mode. While in realm-config mode, you assign a tcp-media-profile to a realm.

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ```

```
ORACLE(media-manager)# realm-config
ORACLE(realm-config)#
```

2. Use the **select** command to identify the target realm.
3. Use the **tcp-media-profile** parameter to assign a specific, named tcp-media-profile to the target realm.

```
ORACLE(realm-config)# tcp-media-profile tlsMutualAuth
ORACLE(realm-config)#
```

4. Use **done**, **exit**, and **verify-config** to complete tcp-media-profile assignment.

tls-profile Configuration

Use the following procedure to create a tls-profile configuration object, which specifies cryptographic resources available in support of TLS operations.

> **Note:**
>
> The option *allow-self-signed-cert* is only available for MSRP connections.

1. Access the **tls-profile** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# security
ORACLE(security)# tls-profile
ORACLE(tls-profile)#
```

2. Use the **name** parameter to provide a unique identifier for this TLS Profile instance.

```
ORACLE(tls-profile)# name tlsMutualAuth
ORACLE(tls-profile)#
```

3. If the require-fingerprint attribute of the tcp-media-profile is set to **enabled**, use the **mutual-authenticate** parameter to enable mutual authentication.

```
ORACLE(tls-profile)# mutual-authenticate enabled
ORACLE(tls-profile)#
```

4. Retain default values for other parameters.
5. Type **done** to save your configuration.
6. Repeat Steps 1 through 5 to configure additional tls-profiles as required.

MSRP Management Monitoring

The **show mbcd** statistics command now displays MSRP flow counts for the current statistics window.

```
ORACLE# show mbcd statistics
16:38:49-145
MBCD Status                -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Sessions        1       1       0          1       1       1
Client Trans           0       0       0          3       3       1
Contexts               2       2       0          3       2       3
Flows                  5       5       0          7       4       7
Flow-Port              0       0       0          0       0       0
```

```
Flow-NAT              5       5       0         11       6       7
Flow-RTCP             0       0       0          0       0       0
Flow-Hairpin          0       0       0          0       0       0
Flow-Released         0       0       0          0       0       0
MSM-Release           0       0       0          0       0       0
Rel-Port              0       0       0          0       0       0
Rel-Hairpin           0       0       0          0       0       0
NAT Entries           5       5       0         11       6       7
Free Ports           12      12       0         14      12      12
Used Ports            0       0       0          2       2       2
Port Sorts            -       -       0          0       0
Queued Notify         0       0       0          0       0       0
MBC Trans             0       0       0          3       3       3
MBC Ignored           -       -       0          0       0
ARP Trans             0       0       0          0       0       0
Relatch NAT           0       0       0          0       0       0
Relatch RTCP          0       0       0          0       0       0
Flow-MSRP             2       2       0          2       2       2
SRTP Only Flows       0       0       0          0       0       0
SRTCP Only Flow       0       0       0          0       0       0
SRTP Collapsed        0       0       0          0       0       0
SRTP Sessions         0       0       0          0       0       0
Flow Rate = 0.0
Load Rate = 0.0
ORACLE#
```

A new command (**show msrp stats**) displays the following cumulative counts:

```
ORACLE# show msrp statistics
PPM_ID_MSRP:
    msrpStatsActiveSessions                    : 0
    msrpStatsMaxActiveSessions                 : 2
    msrpStatsEstablishedSessions               : 2
    msrpStatsProvisionedSessions               : 2
    msrpStatsFinishedSessions                  : 2
    msrpStatsReleasedSessions                  : 2
    msrpStatsFailedSessionsCannotRoute         : 0
    msrpStatsFailedSessionsCannotConnect       : 0
    msrpStatsFailedSessionsFingerprintMismatch : 0
    msrpStatsFailedMessagesCannotBeSent        : 0
    msrpStatsFailedMessagesMalformed           : 0
    msrpStatsSendQFullEvents                   : 0
    msrpStatsSendQCongestedEvents              : 0
    msrpStatsSendQCongestionRelievedEvents     : 0
    msrpStatsStreamRequestsReceived            : 4
    msrpStatsStreamRequestSent                 : 4
    msrpStatsStreamResponsesReceived           : 4
    msrpStatsStreamResponseSent                : 4
    msrpStatsStreamErrorNoTransId              : 0
    msrpStatsStreamErrorNoMsgType              : 0
    msrpStatsStreamErrorNoByteLength           : 0
ORACLE#
```

RCSe TLS/TCP Re-Use Connections

In an RCSe environment the sip-interface reuse-connections option is used to make the Oracle USM retain the TCP/TLS connection established by the endpoint during the registration for all subsequent messages to that endpoint, essentially providing for a persistent connection between the Oracle USM and the user equipment (UE).

Field experience uncovered an implementation deficiency associated with these persistent connections particularly within RCSe deployments. The basic scenario is as follows:

1. The UE registers in a TLS realm on SBC1. SBC1 stores the IP:Port from VIA (and Contact) as alias of the currently established connection.
2. The UE transits to another realm/sip-port (same or different Oracle USM) without previously unregistering or closing the TCP connection with the TLS sip-port on SBC1.
3. UE goes back to the TLS realm in SBC1 and establishes a new connection — same source IP as in Step 1, but a different port as in Step 1.

The problem arises at Step 3. If the Oracle USM has not detected that the TLS connection established in Step 1 has been effectively terminated, it will not update the alias connection to that established in Step 3, but instead continue to attempt to use the Step 1 connection.

This means that the next message from the core side to the UE will fail, since the Oracle USM will attempt to send the message of the dead TLS connection — that is using the IP address:port pair passed in Step 1.

All communications to this UE will fail until it sends the next message to the Oracle USM, when the alias connection will be update to the TLS connection in Step 3.

To resolve this issue, the Oracle USM needs to always update the alias table when it receives a new inbound connection on the configured sip-interface.

Option Configuration Guidelines

The following table lists the full range of options that pertain to TLS/TCP Connection reuse with an emphasis on use in RCSe environments

Option	Connection Behavior
reuse-connections	Use/retain first inbound connection until explicitly closed
reuse-connections=latest	Use the last inbound connection, update the alias for each new connection
reuse-connections=no	Establish new connection at each UE access
NOT CONFIGURED	Equivalent to reuse-connections=no

RCSE TLS/TCP Re-Use Connections Configuration

To configure the reuse-connections option:

1. From Superuser mode, use the following command sequence to navigate to the sip-interface configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# sip-interface
ORACLE(sip-interface)# option reuse-connections=latest
ORACLE(sip-interface)#
```

2. Use **done**, **exit**, and **verify-config** to complete configuration.

23

References and Debugging

ACLI Configuration Elements

The following sections describe the Oracle USM's configuration elements that are unique to S-CZ7.3.5.

sip-registrar

Parameters

name—Configured name of this sip registrar.

- Default: empty

state—Running status of this policy-director-group.

- Default: enabled
- Values: enabled | disabled

domains—List of registration domains that this Oracle USM is responsible for. * means all domains. These domains are compared for an exact match with the domain in the request-uri of the REGISTER message. the wildcard '*' can also be entered as part of this parameter. This is entered as the domains separated by a space in quotes. No quotes required if only one domain is being configured. "+" and "-"are used to add to subtract from the list.

- Default: empty

subscriber-database-method—Protocol used to connect to User Subscriber Database server.

- Default: CX
- Values: CX | DDNS | local

subscriber-database-config—The configuration element that defines the server used for retrieving user subscriber data. For Cx deployments it is a home-subscriber-server name. For ENUM deployments it is an enum-config name.

- Default: empty

authentication-profile—Name of the sip-authentication-profile configuration used to retrieve authentication data when an endpoint is not authenticated.

- Default: empty

home-server-route—The value inserted into the Server Name AVP in an MAR message. This should be entered as a SIP URI as per 3gpp TS 24229 & RFC 3261. The host can be FQDN or IPv4 address, and the port portion should be in the 1025 - 65535 range. Examples: SIP: 12.12.12.12:5060

- Default: empty

third-party-registrars—The third-party-regs configuration element names where third party REGISTER messages will be forwarded to.

- Default: empty

routing-precedence—Indicates whether INVITE routing lookup should use the user database (via the registrar configuration element) or perform local policy lookup immediately.

- Default: registrar
- Values: registrar | local-policy

egress-realm-id—Indicates the default egress/core realm for SIP messaging.

- Default: empty

location-update-interval—Sets the maximum period in minutes in which the core-side user subscriber database is refreshed, per user.

- Default: 1440
- Values: 0-999999999

ifc-profile—References the ifc-profile configuration element's name that is applied to this sip-registrar.

max-contacts-per-aor—Limit to the number of contacts allowed for a given AOR.

- Default: 0 (disabled)
- Values: 1 - 256

Path

This sip-registrar configuration element is a element in the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **sip-registrar**.

sip-authentication-profile

Parameters

name—Configured name of this sip-authentication profile.

methods—List of SIP methods that prompt authentication. This is entered as the methods separated by a space in quotes. No quotes required if only one method is being configured. "+" and "-"are used to add to subtract from the list.

- Default: empty

anonymous-methods—List of SIP methods that prompt authentication when received from anonymous sources. This is entered as the methods separated by a space in quotes. No quotes required if only one method is being configured. "+" and "-" are used to add or subtract from the list.

- Default: empty

digest-realm—The value inserted into the digest-realm parameter in an authentication challenge header as sent to UA. (not used for Cx deployments)

- Default: empty

credential-retrieval-method—Protocol used to connect to the server providing authentication data.

- Default: ENUM-TXT
- Values: ENUM-TXT | CX

credential-retrieval-config—The home-subscriber-server name used for retrieving authentication data.

- Default: empty

Path

This sip-authentication-profile configuration element is a element in the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **sip-authentication-profile**.

home-subscriber-server

Parameters

name—Configured name of this home subscriber server.

- Default: empty

state—Running status of this home subscriber server.

- Default: enabled
- Values: enabled | disabled

transport— The layer 4 protocol used to communicate with this home subscriber server.

- Default: tcp
- Values: tcp | sctp

address—This home subscriber server's IP address.

- Default: none
- Values: IP address in IPv4 or IPv6 format

port—This home subscriber server's port.

- Default: 80
- Values: 1-65535

realm—Oracle USM realm-config name where this home subscriber server exists.

- Default: none

multi-homed-addrs— Specifies one or more local secondary addresses of the SCTP endpoint. This setting is only applicable to SCTP transport. To enter multiple addresses, bracket an address list with parentheses. At least one address is required if transport is set to SCTP.

Multi-homed addresses must be of the same type (IPv4 or IPv6) as that specified by the address parameter. Like the address parameter, these addresses identify SD physical interfaces.

origin-host-identifier—Used to create segment before the dot in the Origin Host AVP.

- Default: none

origin-realm—Populates the value of the Origin Realm AVP. Populates the segment after the dot in the Origin Host AVP.

- Default: none

destination-host-identifier—Used to create segment before the dot in the Destination Host AVP.

- Default: none

watchdog-ka-timer— The interval in seconds of the watchdog/keep-alive messages.

- Default: 0
- Values: 0-65535

num-auth-vector— The number of authentication vectors downloaded from the HSS per MAR.

- Default: 3
- Values: 1-10

Path

This home-subscriber-server configuration element is a element in the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **home-subscriber-server**.

third-party-regs

Parameters

state—Running status of this third party registration configuration element.

- Default: enabled
- Values: enabled | disabled

name—Configured name of this third party registration configuration element.

- Default: none

registrar-host—hostname of the configured session agent that will be third party server. This value is also used in the request-uri that is sent to the third party server.

- Default: none

from-user—The user part of the From URI in the REGISTER Request that is sent to the third party server in the REGISTER message. When this parameter is blank the user part of the From header from the incoming REGISTER Request will be used.

- Default: none

from-host—The host part of the From URI in the REGISTER Request that is sent the third party server in the REGISTER message. When this parameter is blank the Oracle USM uses the egress hostname/ IP address as the host.

- Default: none

- Values: Format this the same as the "registrar-host" in sip-config.

retry-interval—number of seconds the Oracle USM waits before retrying a 3rd Party Registration server after a failed registration.

- Default: 32
- Values: 0 - 3600

Path

This third-party-regs configuration element is a element in the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **third-party-regs**.

local-subscriber-table

Parameters

name—A given name for this local subscriber table element. This name is referenced from the sip-registrar configuration element when the **credential-retrieval-method** is set to **local**.

filename—The filename of local subscriber table that this element references. If no path is provided, the default location is /code/lst.

secret—PSK used for encrypted passwords. This value is not echoed back to the screen upon viewing the configuration element.

Path

The location of this configuration element is: configure terminatl > session-router > local-subscriber-table.

enum-config

Parameters

name—Name for this enum-config to be referenced from within the system.

top-level-domain—The domain extension used to query the ENUM servers for this configuration.

realm-id—The realm-id is used to determine on which network interface to issue an ENUM query.

enum-servers—List of IP address that service the top level domain.

service-type—The ENUM service types you want supported in this ENUM configuration. Possible entries are E2U+sip and sip+E2U (the default), and the types outlines in RFCs 2916 and 3721.

- Default: E2U+sip,sip+E2U

query-method—the ENUM query distribution strategy

- Default: hunt

- Values: hunt | round-robin

timeout—The total time, in seconds, that should elapse before a query sent to a server (and its retransmissions) will timeout.

- Default: 11

cacheInactivityTimer—Enter the time interval, in seconds, after which you want cache entries created by ENUM requests deleted, if inactive for this interval.

- Default: 3600
- Values: 0-999999999

max-response-size—The maximum size in bytes for UDP datagram responses

- Defaults: 512

health-query-number—The phone number for the ENUM server health query; when this parameter is blank the feature is disabled.

health-query-interval—The interval in seconds at which you want to query ENUM server health.

- Default: 0
- Values: 0-65535

failover-to—Name of the enum-config to which you want to failover.

cache-addl-records—Set this parameter to **enabled** to add additional records received in an ENUM query to the local DNS cache.

- Default: enabled
- Values: enabled | disabled

include-source-info—Set this parameter to enabled to send source URI information to the ENUM server with any ENUM queries.

- Default: disabled
- Values: enabled | disabled

ttl—This value ets the TTL value (in seconds) for NAPTR entries in the local ENUM cache and populates when sending a NAPTR entry to the ENUM server.

- Default: 0
- Values: 1-2592000

order—This parameter value populates the order field with when sending NAPTR entries to the ENUM server.

- Default: 1
- Values: 0-65535

preference—This parameter value populates the preference field with when sending NAPTR entries to the ENUM server.

- Default: 1
- Values: 0-65535

Path

This enum-config configuration element is a element in the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **enum-config**.

ifc-profile

Parameters

name—A given name for this IFC profile element. This name is referenced from the sip-registrar configuration element's **ifc-support** parameter.

state—Running status of this IFC profile.

- Default: enabled
- Values: enabled | disabled

shared-ifc-filename—The name of the file referenced for shared IFC function.

default-ifc-filename—The name of the file referenced for default IFC function. This file may be the same as that used for the shared IFC function.

options—Identifies a set of features that vary depending on the configuration element in which they occur and that are enabled by invocation in the **options** parameter. Set the **options** parameter by typing "options", a Space, and then the option name preceded by a plus sign. If you type the option without the plus sign, you will overwrite any previously configured options. To append the new options to this configuration's options list, you must prefix the new option with a plus sign. Prefixing an option with a minus sign removes it from the list of options.

Path

The location of this configuration element is: **configure terminal** > **session-router** > **ifc-profile**.

regevent-notification-profile

Parameters

name—A given name for this registration event notification profile element. This name is referenced from the sip-registrar configuration element.

min-subscription-duration—The amount of time, in seconds, before the subscription expires, unless it is refreshed.

- Default: 3761 seconds
- Values: 180-6000005 seconds

Path

The location of this configuration element is: **configure terminal** > **session-router** > **regevent-notification-profile**.

hss-group

Parameters

name—Enter the name of the hss-group element. This required entry must follow the Name Format, and it must be unique.

state—Enable or disable the hss-group element.

- Default: enabled
- Values: enabled | disabled

origin-host-identifier—Set this to a string for use in constructing a unique Origin Host AVP.

strategy—Select the HSS allocation options for the hss-group. Strategies determine how HSSs will be chosen by this hss-group element.

- Default: hunt
- Values:
  - hunt—Selects HSSs in the order in which they are listed. For example, if the first server is online, all traffic is sent to the first server. If the first server is offline, the second server is selected. If the first and second servers are offline, the third server is selected. When the Oracle USM detects that a higher priority HSS is back in service, it routes all subsequent traffic to that HSS.
  - roundrobin—Selects each HSS in the order in which they are listed in the dest list, selecting each HSS in turn, one per session. After all HSSs have been used, the first HSS is used again and the cycle continues.
  - failover—Selects the first sever in the list until failure is detected. Subsequent signaling goes to the next server in the list.

hss-configs—Identify the home-subscriber-servers available for use by this hss-group. This list can contain as many home subscriber servers as is necessary. An hss-config list value must correspond to a valid hss-group name in another group or to a valid hostname of a configured home-subscriber-server.

A value you enter here must correspond to a valid group name for a configured home-subscriber-server or a valid hostname or IP address for a configured home-subscriber-server.

hss-group is an element under the session-router path. The full path from the topmost ACLI prompt is: **configure terminal** > **session-router** > **session-group**.

SNMP MIBs and Traps

The following MIBs and traps are supported for the Oracle USM. Please consult the Oracle Communications S-CX6.3.0 MIB Reference Guide for more SNMP information.

Acme Packet License MIB (ap-license.mib)

The following table describes the SNMP GET query names for the Oracle License MIB (ap-license.mib).

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apLicenseEntry (1.3.6.1.4.1.9148.3.5.1.1.1)		
apLicenseAuthFeature	apLicenseEntry: 1.3.6.1.4.1.9148.3.5.1.1.1.20	If authorization and authentication is allowed for the Oracle USM, the value is true. If disabled, the value is false.
apLicenseDatabaseRegFeature	apLicenseEntry: 1.3.6.1.4.1.9148.3.5.1.1.1.21	If the Oracle USM is configured as a registrar, the value is true. If registrar functionality is not enabled, this value is false.
apLicenseDatabaseRegCap	apLicenseEntry: 1.3.6.1.4.1.9148.3.5.1.1.1.22	The database registration contact capacity.

Acme Packet System Management MIB (ap-smgmt.mib)

The following table describes the SNMP GET query names for the Oracle System Management MIB (ap-smgmt.mib).

SNMP GET Query Name	Object Identifier Name: Number	Description
Object Identifier Name: apSysMgmtMIBObjects (1.3.6.1.4.1.9148.3.2.1)		
Object Identifier Name: apSysMgmtGeneralObjects (1.3.6.1.4.1.9148.3.2.1.1)		
apSysSipStatsActiveDatabaseContacts	apSysMgmtGeneralObjects: 1.3.6.1.4.1.9148.3.2.1.1.24.0	Number of database-type contacts in the registration cache.

Enterprise Traps

The following table identifies the proprietary traps that Oracle USM system supports.

Trap Name: OID	Description
apSysMgmtDatabaseRegCacheCapTrap: 1.3.6.1.4.1.9148.3.2.6.0.76	Generated when the number of database-type contacts stored in the registration cache exceeds the license threshold.
apSysMgmtDatabaseRegCacheCapClearTrap: 1.3.6.1.4.1.9148.3.2.6.0.77	Trap is generated when the number of database-type contacts stored in the registration cache falls below the license threshold.

Oracle USM Show Commands

show sipd endpoint-ip

The show sipd endpoint-ip <user | IP address> command displays information about each endpoint. For a supplied AoR, the Oracle USM displays all associated contacts (both access and core side), the expiration of each contact entry and associated 3rd Party Registration information. For example:

```
ORACLE# show sipd endpoint-ip 11111
User <sip:111111@172.16.17.100>
  Contact exp=1198
    UA-Contact: <sip:111111@172.16.17.100:5060> UDP keep-acl
         realm=net172 local=172.16.101.13:5060 UA=172.16.17.100:5060
    SD-Contact: <sip:111111-s37q249kvluaa@192.168.101.13:5060> realm=net192
    Call-ID: 1-15822@172.16.17.100'
Third Party Registration:
   Third Party Reg User=<sip:111111@172.16.17.100> state: REGISTERED
   Expire Secs=298 seqNum= 1 refreshInterval=300
   Call-ID: d355a67277d9158e7901e46a12719663@192.168.101.13
   Third Party Reg User=<sip:111111@172.16.17.100> state: REGISTERED
   Expire Secs=178 seqNum= 1 refreshInterval=180
   Call-ID: 07ebbdebfdf64a48985bb82fa8b4c595@192.168.101.13
```

show sipd third-party

The show sipd third-party command displays the current status of third party servers and statistics for messages. The format is:

```
show sipd third-party <all | name>
```

The name argument allows status to be displayed for just the server specified by the name. Not specifying a name results in status being displayed for all third party servers. For example:

```
ORACLE# show sipd third-party-reg all
3rd Party Registrar   SA State  Requests  200OK  Timeouts    Errors
192.168.17.101           INSV         9      9         0         0
192.168.17.102           INSV        14     14         0         0
```

Column definitions are as follows:

- IP Address —IP Address of third party server
- Status —Session Agent State
- Requests —Register requests sent
- 200 OK —200 OK Responses received
- Timeouts —Requests timed out
- Error —Error Responses

show sipd local-subscription

The ACLI **show sipd** command includes an argument that provides information about local subscriptions, as shown below.

```
ORACLE# show sipd local-subscription
19:22:18-152
SIP Local Subscription Status  -- Period -- ----- Lifetime -------
                      Active  High   Total Total  PerMax    High
Server Subscription        0     1       1     1       1       1
Message Statistics
SUBSCRIBE
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
SUBSCRIBE Requests       2          2       2        0          0       0
Retransmissions          0          0       0        0          0       0
200 OK                   1          1       1        0          0       0
403 Forbidden            1          1       1        0          0       0
Response Retrans         0          0       0        0          0       0
Transaction Timeouts     -          -       -        0          0       0
Locally Throttled        -          -       -        0          0       0
Avg Latency=0.000 for 0
Max Latency=0.000
NOTIFY
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
NOTIFY Requests          0          0       0        2          2       2
Retransmissions          0          0       0       10         10      10
200 OK                   0          0       0        1          1       1
Transaction Timeouts     -          -       -        0          0       0
Locally Throttled        -          -       -        0          0       0
Avg Latency=0.000 for 0
Max Latency=0.000
```

You can extend upon this ACLI **show sipd** command to include an argument that provides information about registration event package traffic, as shown below.

```
ORACLE# show sipd local-subscription regevent
19:23:08-103
SIP Local Subscription Status -- Period -- ------ Lifetime -----
                      Active High   Total Total  PerMax    High
Server Subscription        0    1       1     1       1       1
Message Statistics
SUBSCRIBE
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
SUBSCRIBE Requests       2          2       2        0          0       0
Retransmissions          0          0       0        0          0       0
200 OK                   1          1       1        0          0       0
403 Forbidden            1          1       1        0          0       0
Response Retrans         0          0       0        0          0       0
Transaction Timeouts     -          -       -        0          0       0
Locally Throttled        -          -       -        0          0       0
Avg Latency=0.000 for 0
Max Latency=0.000
NOTIFY
                    --------- Server --------   --------- Client --------
Message/Event       Recent      Total  PerMax   Recent      Total  PerMax
                    ------  ---------  ------   ------  ---------  ------
NOTIFY Requests          0          0       0        2          2       2
Retransmissions          0          0       0       10         10      10
200 OK                   0          0       0        1          1       1
Transaction Timeouts     -          -       -        0          0       0
```

```
Locally Throttled        -          -       -        0          0       0
Avg Latency=0.000 for 0
Max Latency=0.000
```

The ACLI **show registration sipd** command includes an argument that provides information about a specific user's registration(s), as shown below.

```
ORACLE# show registration sipd by-user ra1 detailed
User: sip:ra1@apkt.com
  Registered at:  2013-06-05-19:23:40    Surrogate User: false
  Contact Information:
    Contact:
      Name: sip:ra1@apkt.com
      Valid: true
      Challenged: false
      Registered at: 2013-06-05-19:23:40
      Last Registered at: 2013-06-05-19:23:40
      Expire: 3581
      Local expire: 41
      Half: 1781
      Registrar IP: 0.0.0.0
      Transport: UDP
      Secure: false
      Local IP: 192.168.101.62:5060
      User Agent Info:
        Contact: sip:ra1@192.168.13.1:5060
        Realm: net192
        IP: 192.168.13.1:5060
      SD Info:
        Contact: sip:ra1-1cdstqjt90hve@172.16.101.62:5060
        Realm: net172
      Call-ID: 1-28361@192.168.13.1
    Associated URI(s):
      URI: sip:ra1@apkt.com
         Filter Criteria:
           Priority: 0
             Filter: None specified
             Application Server: sip:appserv@apkt.com
    Reg Event Subscriptions Terminated locally:
      Number of Subscriptions: 1
```

Subscriber: appserv<sip:appserv@apkt.com>;tag=1 state=active exp=600114

show registration

The show registration command displays cumulative statistics on all current registrations.

```
ORACLE# show registration
15:35:43-177
SIP Registrations          -- Period -- ----- Lifetime ------
                  Active  High   Total Total  PerMax    High
User Entries           0     0       0     0       0       0
Local Contacts         0     0       0     0       0       0
Via Entries            0     0       0     0       0       0
AURI Entries           0     0       0     0       0       0
Free Map Ports         0     0       0     0       0       0
Used Map Ports         0     0       0     0       0       0
Forwards               -     -       0     0       0       0
Refreshes              -     -       0     0       0       0
Rejects                -     -       0     0       0       0
```

```
Timeouts              -     -       0     0       0       0
Fwd Postponed         -     -       0     0       0       0
Fwd Rejected          -     -       0     0       0       0
Refr Extension        0     0       0     0       0       0
Refresh Extended      -     -       0     0       0       0
ContactsPerAor Reject -     -       0     0       0       0
Surrogate Regs        0     0       0     0       0       0
Surrogate Sent        -     -       0     0       0       0
Surrogate Reject      -     -       0     0       0       0
Surrogate Timeout     -     -       0     0       0       0
HNT Entries           0     0       0     0       0       0
Non-HNT Entries       0     0       0     0       0       0
Database Regs         0     0       0     0       0       0
DDNS Entries          0     0       0     0       0       0
CX Entries            0     0       0     0       0       0
LocalDB Entries       0     0       0     0       0       0
Unreg Users           0     0       0     0       0       0
```

You can extend upon the show registration command by adding the sipd by-user <username> detail arguments. The resulting output reflects user registration information including downloaded IFCs. For example:

```
ORACLE# show registration sipd by-user +19999092907 d
Registration Cache (Detailed View)    MON JUN 25 2012  13:47:46
User: sip:+19999092907@mobile.com
  Registered at: 2012-06-25-13:43:50     Surrogate User: false
  Contact Information:
    Contact:
      Name: sip:+19999092907@mobile.com
      Valid: true
      Challenged: false
      Registered at: 2012-06-25-13:43:50
      Last Registered at: 2012-06-25-13:47:30
      Expire: 48
      Local expire: 13
      Registrar IP: 0.0.0.0
      Transport: UDP
      Secure: false
      Local IP: 155.212.214.175:5060
      User Agent Info:
        Contact: sip:+19999092907@50.76.51.62:5762;transport=udp;acme_nat=
+19999092907+50.76.51.62@10.1.10.20:5762
        Realm: access
        IP: 50.76.51.62:5762
      SD Info:
        Contact: sip:+19999092907-rb8tulsbv3u72@108.108.108.108:5060
        Realm: core
      Call-ID: H_yvkgTAAA@10.1.10.20
    Associated URI(s):
      URI: sip:+19999092907@mobile.com
  Filter Criteria:
    Priority: 0
      Filter: ((case == 'Originating Registered') and (method == INVITE) and
('Accept-Contact'=='+g.app2app')) or
              ((case == 'Originating Registered') and (method == INVITE) and
('Contact'=='+g.app2app')) or
              ((case == 'Originating Registered') and (method == INVITE) and ('P-
Message-Auth'=='.*')) or
              ((case == 'Originating Registered') and (method == INVITE) and ('P-
Application-ID'=='.*'))
      Application Server: sip:pza.mobile.com:5280
```

```
Reg Event Subscriptions Received by Registrar:
Number of Subscriptions : 2
Subscriber: sip:appserv@192.168.13.1:5060; state=active; exp=59978
Subscriber: sip:pcscf@192.168.13.1:5060; state=active; exp=978
```

show home-subscriber-server

The show home-subscriber-server command displays cumulative statistics on all currently configured HSS servers.

show home-subscriber-server [stats <hss-name>| group group-name]

This command allows you to gather a set of information commonly requested by the Oracle TAC when troubleshooting customers.

The show home-subscriber-server command with no arguments displays the status of each HSS as well as the number of transactions and connections per HSS. For example:

```
ORACLE# show home-subscriber-server
Name                Local-Address        Server-Address       Status
hss1                192.168.207.21:45463 192.168.200.232:3872 Up
----------------------------------------
18:53:25-105
HSS Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       1       4      12152       8       1
Server Trans           0       0       0          7       2       1
Connections            1       1       0         53       2       1
```

Note that the Connections statistic indicates the number of connections after successful CER/CEA handshake.

The table below documents the states the

Field	Description
Active	This status is related to HSS failover and load balancing configurations. The diameter connection is up and being used.
Standby	This status is related to HSS failover and load balancing configurations. The diameter connection is up, but is not being used.
Pending	The Oracle USM has sent a CER and is waiting for a CEA response.
Inactive	The Oracle USM has sent a CER but has not received a CEA response.
Down	The Oracle USM is not attempting to establish a connection with the HSS.

Oracle USM reports on each HSS.

The show home-subscriber-server command with the stats argument displays the number of transactions and connections per HSS as well as the number of messages exchanged with all HSS servers per message type. For example:

```
ORACLE# show home-subscriber-server stats
veloster2# show home-subscriber-server stats
Name                Local-Address        Server-Address       Status
hss1                192.168.207.21:45463 192.168.200.232:3872 Up
----------------------------------------
18:55:03-103
HSS Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           1       1       5      12157       8       1
```

```
Server Trans          0       0       0          7       2       1
Connections           1       1       0         53       2       1
                                          ---- Lifetime ----
                                  Recent      Total  PerMax
UAR                                    0          3       1
   SUBSEQ_REG (2002)                   0          3       1
SAR                                    0          6       3
   SUCCESS (2001)                      0          6       3
MAR                                    0          4       2
   SUCCESS (2001)                      0          4       2
LIR                                    0          1       1
   SUCCESS (2001)                      0          1       1
RTR                                    0          1       1
   SUCCESS (2001)                      0          1       1
PPR                                    0          1       1
   SUCCESS (2001)                      0          1       1
CER                                    0         55       3
   SUCCESS (2001)                      0         53       2
DWR                                    5      12088       5
   SUCCESS (2001)                      4      12041       5
   ERR_TIMEOUT                         0         46       1
DWR Recv                               0          5       2
   SUCCESS (2001)                      0          5       2
TCP Failures                           0        267       6
```

By entering the name of a specific HSS as an argument, the ACLI displays all HSS data for that server only. For example:

```
ACMESYSTEM# show home-subscriber-server stats hss1
```

The show home-subscriber-server command with the group argument displays the number of transactions and connections per the HSS group you specify in the command. For example:

```
ORACLE# show home-subscriber-server group hss-group1
display grp hss-group1
HSS Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                0       0       0          0       0       0
Connections            0       0       0          0       0       0
                             ---- Lifetime ----
                     Recent      Total  PerMax
UAR                       0          0       0
SAR                       0          0       0
MAR                       0          0       0
LIR                       0          0       0
RTR                       0          0       0
PPR                       0          0       0
Sent Requests             0          0       0
Sent Req Accepted         0          0       0
Sent Req Rejected         0          0       0
Sent Req Expired          0          0       0
Sent Req Error            0          0       0
Recv Requests             0          0       0
Recv Req Accepted         0          0       0
Recv Req Rejected         0          0       0
HSS  Errors               0          0       0
```

show http-server

The ACLI **show http-server** command provides basic OAuth information as shown below. The command without arguments displays basis statistics on all servers.

```
ORACLE# show http-server
Name               Server-Address              Status
sk                 host.httpsrv.com            Up
sk1                192.168.19.1:8886           Up
sk2                192.168.19.1:8887           Up
sk3                192.168.19.1:8889           Up
12:56:41-184
HTTP Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                0       0       0          0       0       0
Connections            0       0       0          0       0       0
```

You can extend upon this command to get detailed global statistics by adding the stats argument to the end of this command.

```
ORACLE# show http-server stats
Name               Server-Address              Status
sk                 host.httpsrv.com            Up
sk1                192.168.19.1:8886           Up
sk2                192.168.19.1:8887           Up
sk3                192.168.19.1:8889           Up
HTTP Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                1       1       1          1       1       1
Connections            1       1       1          1       1       1
                          ---- Lifetime ----
                     Recent      Total  PerMax
Sent Requests             0          0       0
Sent Req Accepted         0          0       0
Sent Req Rejected         0          0       0
Sent Req Expired          0          0       0
HTTP Errors               0          0       0
```

You can limit this output to a single server by appending the command with the name of that server.

```
ORACLE# show http-server stats http-server1
Name = http-server1
----------------------------------------
Server-Address            Status
192.168.19.1:8886            Up
----------------------------------------
12:56:41-184
HTTP Status                 -- Period -- -------- Lifetime --------
                  Active    High   Total      Total  PerMax    High
Client Trans           0       0       0          0       0       0
Server Trans           0       0       0          0       0       0
Sockets                0       0       0          0       0       0
Connections            0       0       0          0       0       0
---- Lifetime ----
```

```
                    Recent      Total  PerMax
Sent Requests            0          0       0
Sent Req Accepted        0          0       0
Sent Req Rejected        0          0       0
Sent Req Expired         0          0       0
HTTP Errors              0          0       0
```

Session Load Balancer Support

In order to rapidly increase the number of supported endpoints, the Oracle USM can interoperate with the Oracle Communications SLB. When paired with an Oracle Communications SLB, the Oracle USM maintains its ability to function with an HSS or ENUM database.

In addition, the Oracle USM and Oracle Communications SLB pair supports Cx or ENUM based registrations.

To communicate with an Oracle Communications SLB, in addition to all baseline Oracle Communications SBC SIP functionality, the Oracle USM advertises its registration capacity to the Oracle Communications SLB. This value is defined in the SIP interface as the reg cache limit parameter. Since the Oracle USM has a database registrar license, the lower of the two will be advertised to the SLB.

Verify Config

The Oracle USM performs application specific verification checks when you save a config with the save-config ACLI command. These checks are in addition to baseline Oracle USM verification checks.

sip authentication profile (CX)

If session-router > sip-authentication-profile > credential-retrieval-method = CX then confirm

session-router > sip-authentication-profile > credential-retrieval-config value =

any existing session-router > home-subscriber-server configuration > name value

Error

If the above check fails:

1. A WARNING is displayed on the ACLI.
2. An INFO log message is generated.

sip authentication profile (ENUM)

If session-router > sip-authentication-profile > credential-retrieval-method = ENUM-TXT then confirm

session-router > sip-authentication-profile > credential-retrieval-config value =

any existing session-router > enum-config > name value

Error

If the above check fails:

1. A WARNING is displayed on the ACLI.
2. An INFO log message is generated.

sip authentication profile (Local)

If session-router > sip-authentication-profile > credential-retrieval-method = local then confirm

session-router > sip-authentication-profile > credential-retrieval-config =

session-router > local-subscriber-table > ame Error

If the above check fails:

1. A WARNING is displayed on the ACLI.
2. An INFO log message is generated.

sip-registrar

If session-router > sip-registrar > subscriber-database-method = DDNS then confirm

session-router > sip-registrar > subscriber-database-config value =

any existing session-router > enum-config > name value

Error

If the above check fails:

1. A WARNING is displayed on the ACLI.
2. An INFO log message is generated.

sip-registrar

If session-router > sip-registrar > authentication-profile is configured, then confirm its value is any existing:

session-router > sip-authentication-profile > name value

Error

If the above check fails:

1. A WARNING is displayed on the ACLI.
2. An INFO log message is generated.

Resource Utilization

The Oracle USM limits resource utilization to maintain operational stability. Resources managed this way include:

- CPU
- Memory (heap)

CPU Overload Protection

CPU overload protection on the Oracle USM is system-oriented in terms of defining the percent utilization that triggers an action. Actions are application-specific.

For the Oracle USM application, if the CPU usage exceeds the configured setting, the system sends a 5xx error in response to any initial dialog request or standalone transactions. The Oracle USM continues to accept registration refreshes and new transactions within a dialog.

> **Note:**
>
> An Oracle CSM configured to operation as an SLRM rejects all messages when CPU utilization exceeds this threshold.

By default the CPU utilization rate is 80%. This value can be changed by the following ACLI command sequence.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# options +load-limit="70"
ORACLE(sip-config)# done
```

Heap Utilization

The Oracle USM limits memory utilization to maintain operational stability, as follows:

- When heap utilization exceeds the default (75%) or configured memory utilization threshold, the Oracle USM no longer accepts new registrations. The Oracle USM replies to these messages with 5xx messages. The Oracle USM continues to accept registration refreshes, in-dialog calls and subscriptions.
- When heap utilization exceeds its default (90%) or configured threshold, the Oracle USM drops all messages.

The user can change these thresholds to higher or lower values to best accommodate their operational environment. The user can also determine current memory utilization using the following command and referring to the heap utilization value, towards the bottom of the command's output.

```
ORACLE# show platform heap-statistics
```

The user can change the first threshold, for example from its default of 75% to 80%, using the option shown below.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# +options memory-overload-protect 80
ORACLE(sip-config)# done
```

The user can change the default drop-all threshold, from 90% to 85% for example, using the option shown below.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# sip-config
ORACLE(sip-config)# +options heap-threshold 85
ORACLE(sip-config)# done
```

A

RTC Support

This appendix summarizes real-time configuration (RTC) support status for the Oracle USM . The table below lists which configuration elements are supported by RTC and which are not.

ACLI Configuration Elements	Parameter Details
Access Control	
Accounting Config	
Authentication	
Certificate Record	
Class Profile	
Codec Policy	
DNS ALG Service	
DNS Config	
Enum	
External Policy Server	
H.323	The following H.323 stack subelement parameters are not RTC supported in that, if you save and activate a configuration, calls already in progress will be dropped. A reboot is required. • state • isgateway • realm-id • assoc-stack • options • proxy-mode • local-ip • max-calls • max-channels • registration-ttl • terminal-alias • prefixes • ras-port

ACLI Configuration Elements	Parameter Details
	• q931-port • auto-gk-discovery • multicast • gatekeeper • h245-tunneling • gk-identifier • alternate-transport • q931-max-calls • filename
Host Route	
IPSEC	
IWF	
Licensing	
Local Policy	
Local Response Map	
Local Routing Config	
Media Manager	The Media Manager element is supported with the exception of the following parameters: • red-flow-port • red-max-trans • red-sync-start-time • red-sync-comp-time
Media Policy	
Media Profile	
Network Interface	
Net Management Control	
Network Parameters	The Network Parameters element is supported with the exception of the following parameters: • SCTP parameters
NTP Sync	
Packet Trace Config	
Q850 SIP Map	

ACLI Configuration Elements	Parameter Details
Realm Config	
Redundancy Config	The Redundancy Config element is supported with the exception of the following parameters: • state • port • cfg-port • cfg-max-trans • cfg-sync-start-time • cfg-sync-comp-time
Session Agent	
Session Group	
Session Router	
Session Constraints	
Session Translation	
SIP Config	The SIP Config element is supported with the exception of the following parameters: • red-sip-port • red-max-trans • red-sync-start-time • red-sync-stop-time
SIP Feature	
SIP Interface	collect>boot-state
SIP Manipulation	
SIP NAT	The SIP NAT element is supported with the exception of the following parameters: • ext-address
SIP Response Map	
SNMP	
Static Flow	
Steering Pool	
Surrogate Agent	
System	The System Config element is supported with the exception of the following parameters:

ACLI Configuration Elements	Parameter Details
	• options
Test Pattern Rule	
Test Policy	
Test Translation	
TLS Global	When configured to support MSRP over TLS, this configuration element is not RTC enabled. If a single TLS profile is used by both SIP and MSRP, RTC changes will be applied for the SIP TLS configuration ONLY.
TLS Profile	
Translation Rules	
Trap Receiver	

B

USM Base Configuration Elements

This appendix presents base configuration settings required for the following USM deployment types:

- Cx
- ENUM
- LST

USM Base Configuration Elements for Cx

This appendix provides configuration samples of the elements that are required for minimal Oracle USM operation.

The SIP Config must be enabled

```
sip-config
        state                                   enabled
```

You must have a default gateway in your system-config

```
system-config
        default-gateway                          10.0.0.1
```

You must have an access physical interface

```
phy-interface
        name                                    s0p0
        operation-type                          Media
        port                                    0
        slot                                    0
```

You must have a core physical interface

```
phy-interface
        name                                    s0p1
        operation-type                          Media
        port                                    1
        slot                                    0
```

You must have an access network interfaces

```
network-interface
        name                                    s0p0
        sub-port-id                             0
        ip-address                              192.170.1.100
        netmask                                 255.255.255.0
        gateway                                 192.170.1.1
```

You must have a core network interfaces

```
network-interface
        name                                    s0p1
        sub-port-id                             0
        ip-address                              192.170.2.100
        netmask                                 255.255.255.0
        gateway                                 192.170.2.1
```

You must have an access realm

```
realm-config
        identifier                              access1
        addr-prefix                             0.0.0.0
        network-interfaces                      s0p0:0
```

You must have a core realm

```
realm-config
        identifier                              core1
        addr-prefix                             0.0.0.0
        network-interfaces                      s0p1:0
```

You must have an access SIP interface

```
sip-interface
        state                                   enabled
        realm-id                                access1
        sip-port
                address                                 192.170.1.100
                port                                    5060
                transport-protocol                      UDP
                allow-anonymous                         registered
        network-id                              My_Network_Name
        trust-mode                              none
        registration-caching                    enabled
        ims-access                              enabled
```

You must have a core SIP interface

```
sip-interface
        state                                   enabled
        realm-id                                core1
        sip-port
                address                                 192.170.2.100
```

You must have an ENUM Configuration

```
enum-config
        name                                    My_e164_cfg
        realm-id                                core1
        enum-servers                            192.170.2.201
```

You must have a Subscriber Database

```
home-subscriber-server
        name                                    My_HSS
        address                                 192.170.2.202
        realm                                   core1
```

You must have a Registration Event Profile

```
regevent-notification-profile
        name                                    My_reg_event_Profile
```

You must have an Authentication Profile

```
sip-authentication-profile
        name                                    My_Auth_Profile
        methods                                 REGISTER
        anonymous-methods                       *
        digest-realm                            My_Digest_Realm.com
        credential-retrieval-method             Cx
        credential-retrieval-config             My_HSS
```

You must have a Sip-Registrar

```
sip-registrar
        name                                    My_Registrar_Name
        domains                                 my_customer1.com
        subscriber-database-method              Cx
        subscriber-database-config              My_HSS
        authentication-profile                  My_Auth_Profile
        home-server-route                       sip:192.170.2.201:5060
        routing-precedence                      REGISTRAR
        egress-realm-id                         core1
        options                      e164-primary-config=enum:My_e164_Cfg
        regevent-notification-profile           My_reg_event_Profile
```

USM Base Configuration Elements for ENUM

This appendix provides configuration samples of the elements that are required for minimal Oracle USM operation.

The SIP Config must be enabled

```
sip-config
        state                                   enabled
```

You must have a default gateway in your system-config

```
system-config
        default-gateway                          10.0.0.1
```

You must have an access physical interface

```
phy-interface
        name                                    s0p0
        operation-type                          Media
        port                                    0
        slot                                    0
```

You must have a core physical interface

```
phy-interface
        name                                    s0p1
        operation-type                          Media
        port                                    1
        slot                                    0
```

You must have an access network interfaces

```
network-interface
        name                                    s0p0
        sub-port-id                             0
        ip-address                              192.170.1.100
        netmask                                 255.255.255.0
        gateway                                 192.170.1.1
```

You must have a core network interfaces

```
network-interface
        name                                    s0p1
        sub-port-id                             0
        ip-address                              192.170.2.100
        netmask                                 255.255.255.0
        gateway                                 192.170.2.1
```

You must have an access realm

```
realm-config
        identifier                              access1
        addr-prefix                             0.0.0.0
        network-interfaces                      s0p0:0
```

You must have a core realm

```
realm-config
        identifier                              core1
        addr-prefix                             0.0.0.0
        network-interfaces                      s0p1:0
```

You must have an access SIP interface

```
sip-interface
        state                                   enabled
        realm-id                                access1
        sip-port
                address                                 192.170.1.100
                port                                    5060
                transport-protocol                      UDP
                allow-anonymous                         registered
        trust-mode                              none
        registration-caching                    enabled
        ims-access                              enabled
```

You must have a core SIP interface

```
sip-interface
        state                                   enabled
        realm-id                                core1
        sip-port
                address                                 192.170.2.100
```

You must have an ENUM Configuration for e.164 translation

```
enum-config
        name                                    My_e164_cfg
        realm-id                                core1
        enum-servers                            192.170.2.201
```

You must have an ENUM Configuration for accessing ENUM server(s) as subscriber server(s)

```
enum-config
        name                                    My_ENUM_servers
        realm-id                                core1
        enum-servers                            192.170.2.203
        cacheInactivityTimer                    0
        max-response-size                       65535
```

You must have an Authentication Profile

```
sip-authentication-profile
        name                                    My_Auth_Profile
        methods                                 REGISTER
        anonymous-methods                       *
        digest-realm                            My_Digest_Realm.com
        credential-retrieval-method             enum-text
        credential-retrieval-config             My_ENUM_servers
```

You must have a Sip-Registrar

```
sip-registrar
        name                                    My_Registrar_Name
        domains                                 my_customer1.com
        subscriber-database-method              DDNS
        subscriber-database-config              My_ENUM_servers
        authentication-profile                  My_Auth_Profile
        home-server-route                       sip:192.170.2.201:5060
        routing-precedence                      REGISTRAR
        egress-realm-id                         core1
        options                     e164-primary-config=enum:My_e164_Cfg
```

USM Base Configuration Elements for LST

This appendix provides configuration samples of the elements that are required for minimal Oracle USM operation.

The SIP Config must be enabled

```
sip-config
        state                                   enabled
```

You must have a default gateway in your system-config

```
system-config
        default-gateway                          10.0.0.1
```

You must have an access physical interface

```
phy-interface
        name                                     s0p0
        operation-type                           Media
        port                                     0
        slot                                     0
```

You must have an access network interfaces

```
network-interface
        name                                     s0p0
        sub-port-id                              0
        ip-address                               192.170.1.100
        netmask                                  255.255.255.0
        gateway                                  192.170.1.1
```

You must have an access realm

```
realm-config
        identifier                               access1
        addr-prefix                              0.0.0.0
        network-interfaces                       s0p0:0
```

You must have an access SIP interface

```
sip-interface
        state                                    enabled
        realm-id                                 access1
        sip-port
                address                                  192.170.1.100
                port                                     5060
                transport-protocol                       UDP
                allow-anonymous                          registered
        trust-mode                               none
        registration-caching                     enabled
        ims-access                               enabled
```

You must have a Local Subscriber Table

```
local-subscriber-table
        name                                     My_LST
        filename                                 /code/lst/My_LST.xml
        secret                                   *******
```

You must have an Authentication Profile

```
sip-authentication-profile
        name                                     My_Auth_Profile
        methods                                  REGISTER
        anonymous-methods                        *
        digest-realm                             My_Digest_Realm.com
        credential-retrieval-method              local
        credential-retrieval-config              My_LST
```

You must have a Sip-Registrar

```
sip-registrar
        name                                    My_Registrar_Name
        domains                                 my_customer1.com
        subscriber-database-method              LOCAL
        subscriber-database-config              My_LST.xml
        authentication-profile                  My_Auth_Profile
        routing-precedence                      REGISTRAR
```