# Oracle® Communications Unified Session Manager

## Call Traffic Monitoring Guide

Release S-CZ7.3.5

October 2018

Oracle Communications Unified Session Manager Call Traffic Monitoring Guide, Release S-CZ7.3.5

# Contents

# About this Guide

The Oracle USM Call Traffic Monitoring Guide provides information about monitoring the call traffic on your system.

### 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 3820 Hardware Installation Guide | Contains information about the components and installation of the Acme Packet 3820. |
| 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. |
| 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. |

| Document Name | Document Description |
|---|---|
| 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. |

### Revision History

| Date | Description |
|---|---|
| March 2016 | • Initial Release |
| February 2017 | • Updates Egress RTP packet capture information under Packet Trace |
| April 2017 | • Clarifies that local packet capture does not include RTP |
| October 2018 | • Adds the CPU load "Note" to the "Oracle Communications Operations Monitor Configuration" and "Configure the Oracle Communications Operations Monitor" topics.<br>• Adds the "Anonymize Sensitive Data in SIP Messages" and "Enable Anonymization" topics. |

1

# Call Traffic Monitoring

## SelectiveCall Recording SIPREC

The SIPREC protocol is the protocol used to interact between a Session Recording Client (SRC) (the role performed by Oracle USM) and a Session Recording Server (SRS) (a 3rd party call recorder or Oracle Communications Interactive Session Recorder's Record and Store Server (RSS)). It controls the recording of media transmitted in the context of a communications session (CS) between multiple user agents.

SIPREC provides a selective-based call recording solution that increases media and signaling performance on 3rd party call recording servers, more robust failovers, and the ability to selectively record.

> **Note:**
>
> SIPREC isolates the 3rd party recorders from the communication session. The 3rd party recorders can determine whether or not recording is desired.

> **Note:**
>
> The SRC starts a recording session for every call within a configured realm. All call filtering, if desired, must be accomplished by the SRS. The SRS performs the filtering and selection of which sessions it should record.

## SIPREC Feature

The SIPREC feature supports active recording, where the Oracle USM acting as the SRC, purposefully streams media to the Oracle Communications Interactive Session Recorder's RSS (or 3rd party call recorder) acting as the SRS. The SRC and SRS act as SIP User Agents (UAs). The SRC provides additional information to the SRS to describe the communication sessions, participants and media streams for the recording session to facilitate archival and retrieval of the recorded information.

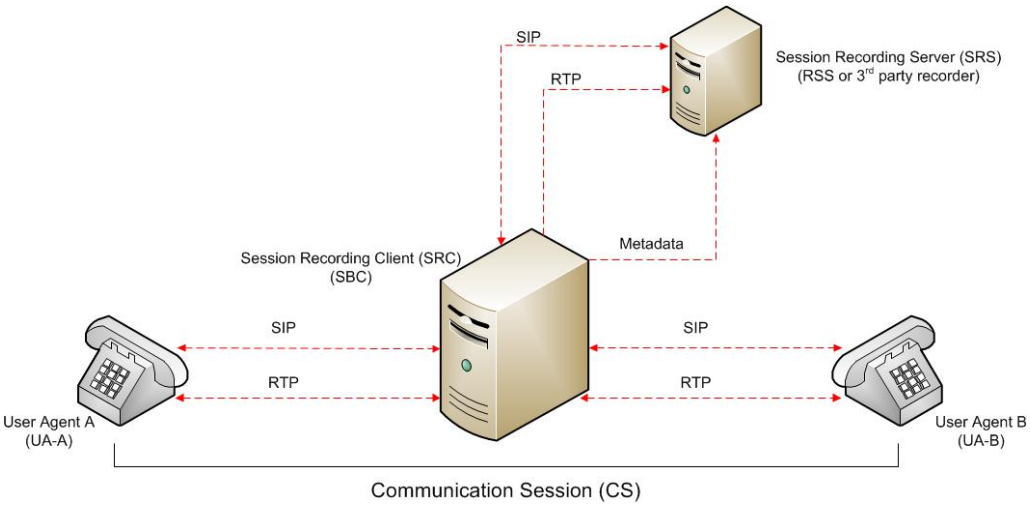
The Oracle USM acting as the SRC, is the source for the recorded media. The Oracle USM consumes configuration information describing the ecosystem within which it operates. The interface, realm and session agent configuration objects specify the SIPREC configuration. A SIP UA can elect to allow or disallow any network element from recording its media.

During the establishment of a SIP Session, the Oracle USM determines if SIPREC is configured for recording the call. If so, it then duplicates the media prior to initiating the session with the SRS. (Media replication is set up prior to the recording session). The SRS may choose to record, not record, or cancel the recording session, and then communicates via SIP

signaling to the Oracle USM. If the call is not to be recorded, the SRS signals termination of the recording session.

The Oracle USM maintains SIPREC metadata information associated with recording sessions. The recording session metadata describes the current state of the recording session and its communication session(s). It is updated when a change of state in the communication session(s) is observed by the Oracle USM. The SRS is responsible for maintaining call history, etc. The Oracle USM creates and logs call detail records (CDRs) in the current manner, the 3rd party SRS vendor may collate this information if desired. (For more information about the contents of metadata, see Metadata Contents).

The following illustration shows two endpoints, User Agent A (UA-A) and User Agent B (UA-B). Their session is being recorded by an SRC (the Oracle USM) and an SRS.




# Configuring SIPREC

This section defines the information required to configure SIPREC on the Oracle USM. It also provides a sample procedure for configuring SIPREC using the Acme Packet Command Line Interface (ACLI).

## Session Recording Server (SRS)

The Oracle Communications Interactive Session Recorder's RSS acts as the SRS in the network. A **session-recording-server** attribute under the **session-router** object in the Oracle USM ACLI allows you to enable/disable the SRS. This object is the session recording server that receives replicated media and records signaling. Additional parameters for SRS are configured under the **session-agent**, **realm-config**, and **sip-interface** objects. The rules of precedence for which the Oracle USM uses these parameters are:
**session-agent** takes precedence over the **realm-config**, and **realm-config** takes precedence over **sip-interface**.

Each SRS is associated with a **realm-config**. The realm specifies the source interface from which replicated traffic originates. The destination is an IP Port parameter (IP address or hostname with an optional port) that defines the SIP address (request URI) of the actual SRS.

For an additional level of security, Oracle recommends the SRS be configured in its own realm so as to apply a set of access control lists (ACLs) and security for the replicated communication.

Although the Oracle USM supports large UDP packets, Oracle recommends the **sip-interface** associated with the SRS realm, be provisioned with a TCP port.

# Session Recording Group

The Oracle USM uses the **session-recording-group** attribute under the **session-router** object in the ACLI to set high availability (HA) for 3rd party call recorders. Using this object, you can define a collection of one or more SRSs. The Oracle USM utilizes SIP's transport mechanism and keeps track of statistics on each SRS to manage the distribution of traffic and load balancing. (For more information on Oracle USM load balancing in session recording groups, see Load Balancing). When multiple SRSs are in a session recording group, the Oracle USM uses heuristics to intelligently route the recording dialog to one or more SRSs utilizing the selection strategy.

The **simultaneous-recording-servers** configuration attribute controls the number of simultaneous SIP dialogs that the Oracle USM establishes to the SRSs in the session recording group per communication session. For instance, if a session recording group contains 3 SRSs, and **simultaneous-recording-servers** is set to **2**, the recording agent initiates a SIP INVITE to the next two SRSs based on the session recording group strategy. In this way, duplicative recording sessions are instantiated, allowing for recording redundancy in multiple SRSs or within a session recording group.

> **Note:**
>
> The Oracle USM streams media to all SRSs. Each SRS chooses whether or not to ignore the media by returning a recvonly(receive only) media line. This permits an SRS to select specific media to record in the recording session, as well as determine whether or not to record the media.

The number of simultaneous recording servers does not dictate the number of recording devices required to be active for a communication session. If two SRSs exist in a session recording group and **simultaneous-recording-servers** is set to **2**, if at least one recording device to any of the servers completes, the recording server is treated as being established.

# Load Balancing

The Oracle USM supports recording server load balancing across members of a session recording group using the following strategies:

> **Note:**
>
> SRS groups support "round-robin" and "hunt" strategies only.

[**Round-robin**]: The Oracle USM remembers the last SRS that was used. Each new recording session selects the next SRS in the session recording group. When simultaneous-recording-servers is greater than 1, the next n recording servers are selected from the session recording group.

[**hunt**]: The Oracle USM successively attempts to contact SRSs in the session recording group until a successful recording dialog is established with the SRS, starting from the first SRS in the session recording group. The Oracle USM attempts to contact each SRS in the session reporting group once. When contact is exhausted, the recording device is considered failed. A SIP failure (response greater than 399, timeout or TCP setup failure) causes the Oracle USM to attempt the next possible SRS. When simultaneous-recording-servers is greater than 1, the Oracle USM attempts to establish n recording devices in a hunting fashion.

## Session Recording Group within Logical Remote Entities

Each logical remote entity (session-agent, realm-config and sip-interface) has a **session-recording-server attribute.**This attribute is a reference to a specific SRS configuration and can be used to specify a session recording group instead. If a session recording group is specified instead of an SRS, the session recording group name must be prefixed with "**SRG:**" followed by the session recording group name. This distinguishes between an SRS being referenced and a session recording group being referenced.

With SIPREC, if an SRS or session recording group is configured on both the ingress and egress logical remote entities, both the ingress and egress SRS/session recording groups are used. This means that the Oracle USM records the media between participants twice (or more) - once for the ingress recorders and once for the egress recorders.

If both the ingress and egress SRS/session recording group are the same, the Oracle USM makes an optimization and only records the media once. Even if the ingress session recording group is the same exact set of SRSs as the egress session recording group (but with a different name), the Oracle USM replicates media to both destinations. However, if the same set of SRSs has the exact same identifier, the
Oracle USM sends media to one and not both SRSs.

# Selective Recording

SIPREC defines a number of use cases for which the Oracle USM can record communication sessions. These use cases include the use of selective based recording. A **selective recording** is one in which a unique recording server is created per communication session.

> **Note:**
>
> The Oracle USM does not support persistent recording.

For SRSs using selective recording, recording servers are unique per session recording group. For each selective SRS in a session recording group, during the setup of a new communication session, the recording metadata is the same for each recording device. The SRC initiates a new SIP INVITE to the SRS carrying the metadata for that new recording server. The recording agent terminates the SIP dialog at the time that the recording session ends.

The lifetime of a recording session extends beyond the lifetime of the recorded communication. The SRC (Oracle USM) re-uses the recording session ID in the metadata instead of creating a new ID for each recording.

# High Availability (HA) Support

An Oracle USM using SIPREC supports HA in the network. The Oracle USM replicates all metadata states between the active and standby Oracle USMs. Any recording dialogs in

progress do not survive the failover, but all calls in progress are preserved. Additionally, the recording dialogs are replicated as well to the failed over Oracle USM so that in-dialog SIP requests continue to function.

Each recorded communication session replicated to a single SRS counts as two calls instead of one. The Oracle USM creates two flows between the two participants and two additional flows to the SRS for each of the parent flows.

# SIPREC Configuration Procedure

The following configuration example assumes the Oracle USM has the session recording license enabled on the Oracle USM. Changes to the call session recording configuration for SIPREC are dynamic. Active calls in progress remain unaffected by the configuration changes. New calls, however, utilize the changes after a **Save** and **Activate** of the configuration.

The following attributes must be configured:

- session-recording-server
- **session-recording-group** (for RSS or 3rd party SRS high availability (HA) only)

  and at least one of the following attributes:
- realm-config
- session-agent
- sip-interface

## Session-recording-server Attribute

To configure the session-recording-server attribute:

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

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router-related objects.

   ```
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)#
   ```

3. Type **session-recording-server** and press Enter to access the session recording server-related attributes.

   ```
   ACMEPACKET(session-router)# session-recording-server
   ACMEPACKET(session-recording-server)#
   ```

4. **name** — Enter a unique name for the session recording server. This name can be referenced when configuring realm-config, session-agent, and sip-interface. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-server)# name SRS1
   ```

5. **(optional) description** — Enter a description for the session recording server. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-server)# description <recording server name>
   ```

6. **realm** — Enter the realm for which the session recording server belongs. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-server)# realm <realm name>
   ```

> **Note:**
>
> Oracle recommends that the session recording server be configured in its own realm.

7. **mode** — Enter the recording mode for the session recording server. Valid values are:
   - **selective** (default) - Unique recording server created per communication session
   - **persistent** - Not supported.

   ```
   ACMEPACKET(session-recording-server)# recording-mode selective
   ```

8. **destination** — Enter the destination IP address with IP port (port specification is optional) that defines the SIP address (request URI) of the session recording server. Enter values in the format 0.0.0.0:<port number>. Default is no value specified.

   ```
   ACMEPACKET(session-recording-server)# destination 172.34.2.3:5060
   ```

9. **port** — Enter the port number to contact the session recording server. Valid values are 1024 to 65535. Default is 5060.
10. **transport-method** — Enter the protocol that the session recording server uses to accept incoming packets from the session reporting client on the network. Default is DynamicTCP. Valid values are:
    - "" - No transport method used. Same as leaving this parameter value blank.
    - UDP - User Datagram Protocol (UDP) is used for transport method.
    - UDP+TCP - UDP and Transmission Control Protocol (TCP) are used for transport method.
    - DynamicTCP - One TCP connection for EACH session is used for the transport method.
    - StaticTCP - Only one TCP connection for ALL sessions is used for the transport method. This option saves resource allocation (such as ports) during session initiation.
    - DynamicTLS - One Transport Layer Security (TLS) connection for EACH session is used for the transport method.
    - StaticTLS - Only one TLS connection for ALL sessions is used for the transport method. This option saves resource allocation (such as ports) during session initiation.
    - DTLS - Datagram TLS is used for the transport method.
    - TLS+DTLS - TLS and DTLS are used for the transport method.
    - StaticSCTP - Only one Stream Control Transmission Protocol (SCTP) connection for ALL sessions is used for the transport method. This option saves resource allocation (such as ports) during session initiation.

      ```
      ACMEPACKET(session-recording-server)# protocol UDP
      ```

11. Enter **done** to save the session recording configuration.

    ```
    ACMEPACKET(session-recording-server)# done
    ```

12. Enter **exit** to exit the session-recording-server configuration.

    ```
    ACMEPACKET(session-recording-server)# exit
    ```

13. Enter **exit** to exit the session-router configuration.

    ```
    ACMEPACKET(session-router)# exit
    ```

14. Enter **exit** to exit the configure mode.

    ```
    ACMEPACKET(configure)# exit
    ```

15. Enter **save-config** to save the session recording configuration.

    ```
    ACMEPACKET# save-config
    ```

16. Enter **activate-config** to activate the session recording configuration.

    ```
    ACMEPACKET# activate-config
    ```

## Session-recording-group Attribute (for HA only)

For environments that required high availability (HA) requirements, configure the **session-recording-group** attribute.

To configure the session-recording-group attribute and enable HA:

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

   ```
   ACMEPACKET# configure terminal
   ```

2. Type **session-router** and press Enter to access the session router-related objects.

   ```
   ACMEPACKET(configure)# session-router
   ACMEPACKET(session-router)#
   ```

3. Type **session-recording-group** and press Enter to access the session recording group-related attributes.

   ```
   ACMEPACKET(session-router)# session-recording-group
   ACMEPACKET(session-recording-group)#
   ```

4. **name** — Enter a unique name for the session recording group that is a collection of one or more session recording servers. This name can be referenced when configuring realm-config, session-agent, and sip-interface. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-group)# name <SRG Group Name>
   ```

   > **Note:**
   >
   > The name of the session recording group must be prefixed with SRG.

5. **(optional) description** — Enter a description for the session recording group. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-group)# description <Recording Group Name>
   ```

6. **session-recording-servers** — Enter the names of the session recording servers that belong to this session recording group. Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(session-recording-group)# session-recording-servers SRS1,SRS2
   ```

   > **Note:**
   >
   > You must enter multiple servers as values for the session-recording-servers attribute.

7. **strategy** — Enter the load balancing strategy that the session reporting client (Oracle USM) uses when sending recordings to the session reporting server. Valid values are:

   - **Round-robin** (default) - The Oracle USM remembers the last SRS that was used. Each new recording session selects the next SRS in the session recording group. When simultaneous-recording-servers is greater than 1, the next n recording servers are selected from the session recording group.
   - **hunt** - The Oracle USM successively attempts to contact SRSs in the session recording group until a successful recording dialog is established with the SRS, starting from the first SRS in the session recording group. The Oracle USM attempts to contact each SRS in the session reporting group once. When contact is exhausted, the recording device is considered failed. A SIP failure (response greater than 399, timeout or TCP setup failure) causes the Oracle USM to attempt the next possible SRS. When simultaneous-recording-servers is greater than 1, the Oracle USM attempts to establish n recording devices in a hunting fashion.
   - **least busy** - For some 3rd party recording devices, the number of concurrent recording servers proves to be the most taxing for system resources. The Oracle USM tracks the number of recording servers active to a given SRS at any given time. It uses this information to determine which SRS would be the best candidate for the next RS. The SRS with the fewest number of active recording servers receives the next RS. If two or more SRSs in a session recording group currently have the same number of active recording servers, the SRS configured first in the session recording group takes precedence.
   - **lowest sustained rate** (fewest-setups-per-minute) - For some 3rd party recording servers, processing large amounts of sessions in a short amount of time proves to be the most taxing on their system's resources. The Oracle USM tracks the number of recording server setups over a sliding window of five minutes. The SRS within the session recording group with the fewest setups per the window of time is selected as the next candidate for receiving the recorded session. If two or more SRSs in a session recording group currently have the same value for setups in the given window of time, then the SRS configured first in the session recording group takes precedence.

   ```
   ACMEPACKET(session-recording-group)# strategy round-robin
   ```

8. **simultaneous-recording-servers** — Enter the number of simultaneous SIP dialogs that the session reporting client (Oracle USM) establishes to the session reporting servers in the session reporting group per communication session. Valid values are **1** to **3**. Default is **0**.

   ```
   ACMEPACKET(session-recording-group)# simultaneous-recording-servers 2
   ```

9. Enter **done** to save the session recording group configuration.

   ```
   ACMEPACKET(session-recording-group)# done
   ```

10. Enter **exit** to exit the session recording group configuration.

    ```
    ACMEPACKET(session-recording-group)# exit
    ```

11. Enter **exit** to exit the session-router configuration.

    ```
    ACMEPACKET(session-router)# exit
    ```

12. Enter **exit** to exit the configure mode.

    ```
    ACMEPACKET(configure)# exit
    ```

13. Enter **save-config** to save the session recording group configuration.

    ```
    ACMEPACKET# save-config
    ```

14. Enter **activate-config** to activate the session recording group configuration.

```
ACMEPACKET# activate-config
```

## Realm-config Attribute

Use the following procedure to configure the realm-config attribute and enable session recording:

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

   ```
   ORACLE# configure terminal
   ORACLE(configure)# media-manager
   ORACLE(media-manager)# realm-config
   ORACLE(realm-config)#
   ```

2. **session-recording-server** — Enter the name of the session-recording server or the session-recording-group in the realm associated with the session reporting client (Oracle USM). Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ACMEPACKET(realm-config)# session-recording-server <srs-name>

   or

   ACMEPACKET(realm-config)# session-recording-server SRG:<group-name>
   ```

   > **Note:**
   >
   > The value for this attribute is the name you specified in the session-recording-server attribute. If specifying a session-recording-group, you must precede the group name with "SRG:".

3. **session-recording-required** — Enter whether you want a call to be accepted by the Oracle USM when recording is not available. The default value is **disabled**.

   - **Enabled** — Restricts call sessions from being initiated when a recording server is not available.
   - **Disabled** (default) — Allows call sessions to initiate even when the recording server is not available.

   > **Note:**
   >
   > Oracle recommends that the **session-recording-required** parameter remain disabled.

4. **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).

   test

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

## Session-agent Attribute

To configure the session-agent attribute and enable session recording:

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
   ACMEPACKET(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. **session-recording-server** — Enter the name of the session-recording server or the session-recording-group to apply to the session recording client (Oracle USM). Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ORACLE(session-agent)# session-recording-server <srs-name>
   ```

   or

   ```
   ORACLE(session-agent)# session-recording-server SRG:<group-name>
   ```

   > **Note:**
   >
   > The value for this attribute is the name you specified the session-recording-server attribute. If specifying a session-recording-group, you must precede the group name with **SRG:**.

5. **session-recording-required** — Enter whether or not you want a call to be accepted by the Oracle USM if recording is not available. Valid values are:

   - **Enabled** - Restricts call sessions from being initiated when a recording server is not available.
   - **Disabled** (default)- Allows call sessions to initiate even if the recording server is not available.

   ```
   ORACLE(session-agent)# session-recording-required disabled
   ```

   > **Note:**
   >
   > Oracle recommends that the session-recording-required parameter remain disabled.

6. Enter **exit** to exit the session agent configuration.

   ```
   ORACLE(session-agent)# exit
   ```

7. Enter **exit** to exit the session router configuration.

   ```
   ORACLE(session-router)# exit
   ```

8. Enter **exit** to exit the configure mode.

   ```
   ORACLE(configure)# exit
   ```

9. Enter **save-config** to save the session agent configuration.

   ```
   ORACLE# save-config
   ```

10. Enter **activate-config** to activate the session agent configuration.

```
ORACLE# activate-config
```

## Sip-interface Attribute

To configure the sip-interface attribute and enable session recording:

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 **sip-interface** and press Enter to access the SIP interface-related attributes.

   ```
   ORACLE(session-router)# sip-interface
   ORACLE(sip-interface)#
   ```

4. **session-recording-server** — Enter the name of the session-recording server or the session-recording-group to apply to the SIP interface on the session recording client (Oracle USM). Valid values are alpha-numeric characters. Default is no value specified.

   ```
   ORACLE(sip-interface)# se ss on-recording-server SRG:<session recording
   server name or session-recording group name>
   ```

   > **Note:**
   >
   > The value for this attribute is the name you specified in the session-recording-server attribute.

5. **session-recording-required** — Enter whether or not you want a call to be accepted by the Oracle USM if recording is not available. Valid values are:

   - **Enabled** - Restricts call sessions from being initiated when a recording server is not available.
   - **Disabled** (default)- Allows call sessions to initiate even if the recording server is not available.

   ```
   ORACLE(sip-interface)# session-recording-required disabled
   ```

   > **Note:**
   >
   > Oracle recommends that the session-recording-required parameter remain disabled.

6. Enter **exit** to exit the SIP interface configuration.

   ```
   ORACLE(sip-interface)# exit
   ```

7. Enter **exit** to exit the session router configuration.

   ```
   ORACLE(session-router)# exit
   ```

8. Enter **exit** to exit the configure mode.

   ```
   ORACLE(configure)# exit
   ```

9. Enter **save-config** to save the SIP interface configuration.

```
ORACLE# save-config
```

10. Enter **activate-config** to activate the SIP interface configuration.

```
ORACLE# activate-config
```

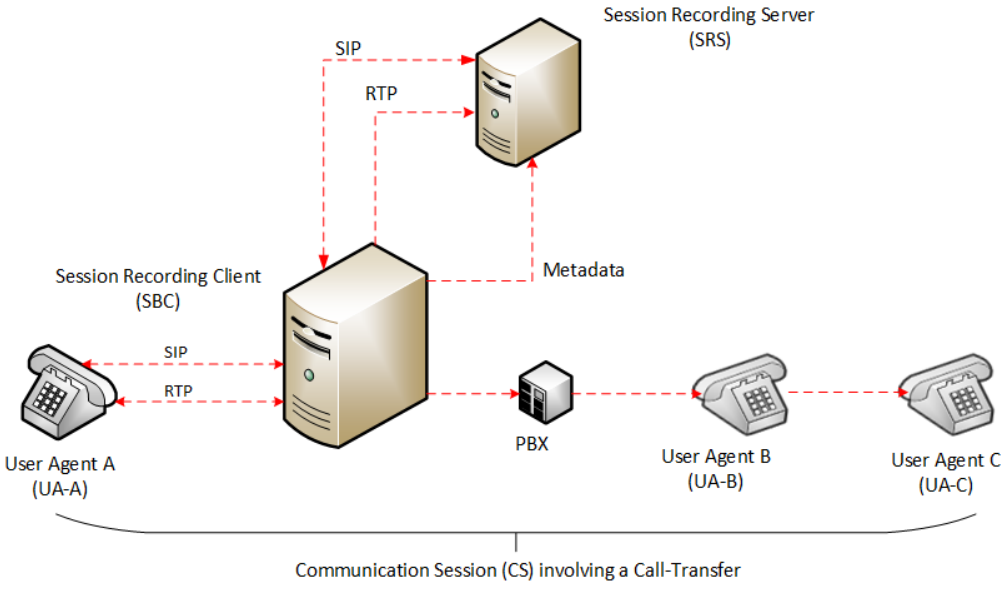
# P-Asserted Identity and Diversion Headers in SIPREC Metadata

The Oracle USM supports some call transfer scenarios in which the contents of the P-Asserted-Identity, Diversion and History-info headers must be included in the SIPREC metadata for in-dialog requests (re-INVITE and UPDATE) as well as initial requests.

During a Communication Session (CS) between user-agents, Oracle USMupdates the SRS, for in-dialog requests like Re-INVITE and UPDATE, in the Recording Session (RS). In call transfer scenarios, the participant information available in the headers is updated in the extension data under SIPREC Metadata. When the option **disable-re-invite-on-update** is configured in the session-agent, sip-interface, or real-config configuration elements, the SBC restricts the Re-INVITE to RS for the in-dialog UPDATE in CS.

You will have to configure the system with the SipHeaderExtensionMetadata.spl (SBC Processing Language) plugin for the **disable-re-invite-on-update** option to work. For more information, see Inserting SIP headers into SIPREC Metadata.

Consider a call transfer scenario between User Agent A (UA-A) and User Agent C (UA-C). The session will be recorded by Oracle USMacting as an SRC and an SRS. The SRS does not record the session with User Agent B. In order to record the session between UA-A and UA-C it is vital to correctly identify the User Agents in the session. The system uses the headers in the metadata to reflect the identities of the participating agents.




The header information such as P-Asserted Identity, Diversion and History-info for INVITE/Re-INVITE and UPDATE cases will be updated in the extension data of participant xml element.

You can configure the option **disable-re-invite-on-update** to stop the Re-INVITE requests towards the Recording Session (RS) for any in-dialog UPDATE in Communication Session (CS). This option can be configured under a specified **sip-interface** , **realm-config** , or a **session-agent** .

```
ORACLE#(session-agent) options +disable-re-invite-on-update
```

**Session-agent** takes precedence over **realm-config**, and **realm-config** takes precedence over sip-interface.

# Metadata Contents

The recording metadata contains a set of related elements which define the recording session. A recording session may contain zero or more communication sessions and/or communication session groups. A communication session represents a call instance; a communication session group represents a related group of communication sessions. A recording session is composed of a sequence of complex element types. Not all element types are required to describe a recording session initiated from the Oracle USM. The recording session XML schema defines the following element types:

- **dataMode** - partial or complete metadata description (required)
- **group** - a collection of related communication sessions
- **session** - a single communication session of two or more participants (required)
- **participant** - a SIP endpoint representation (required)
- **stream** - a media stream
- **extensiondata** - application specific data outside of the SIPREC scope.

The recording agent generates dataMode, session, participant, and stream elements. Extension data is attached to other elements within the metadata through the use of the parent attribute. The recording metadata is defined as a sequence of element types; therefore all associations between elements are represented as references to element identifiers.

The state of the metadata within a recording session reflects the state of the communication session(s) which is being recorded. SIPREC implements stop-times and reason codes when communication sessions end within a recording session. Once a communication session, participant, or media stream has been marked as 'stopped' and accepted by the SRS, the metadata item is removed from the current metadata state. In addition, media lines within the SDP or the recording session may be re-used/re-labeled for reuse if new communication sessions and media streams are created within the recording session.

The XML schema for the recording metadata is defined in the IETF draft RFC *draft-ram-siprec-metadata-format-02* [7].

The ACLI command to show recorded metadata is **show rec**. For more information on this command see the section, Show rec.

# Show Commands for Recording Sessions

The Oracle USM allows you to utilize the following **show** commands via the ACLI to display statistical information about recording sessions:

- show rec
- show rec redundancy

# Show rec

The **show rec** command displays the count of all metadata objects in sessions managed by the recording agent. These statistics include metadata monitored over an active **period** of time and over a lifetime period (where lifetime totals reflect from the last reboot of the Oracle USM to the present time). The following example shows the use of this command.

1. Log into the Oracle USM as a User or Superuser.

   ```
   ACMEPACKET> enable
   ACMEPACKET(enable)#
   ```

2. Type **show rec** and press Enter to display the recording metadata statistics. The following output is an example of the show rec command.

   ```
   ACMEPACKET(enable)# show rec
   ```

   **Show rec output**

   ```
   13:49:44-81645
   Recording Agent Status        -- Period -- -------- Lifetime --------
                   Active    High   Total      Total  PerMax    High
   Rec Sessions         0       1       1          1       1       1
   Comm Groups          0       0       0          0       0       0
   Comm Sessions        0       1       1          1       1       1
   Media Streams        0       2       2          2       2       2
   Participants         0       2       2          2       2       2
   ```

   The following table describes the metadata objects in the show rec command output.

| Object | Description |
|---|---|
| Rec Sessions | Number of recording sessions during an active period of time and over a lifetime period. |
| Comm Groups | Number of active communication session recording groups during an active period of time and over a lifetime period. |
| Comm Sessions | Number of active communication sessions during an active period of time and over a lifetime period. |
| Media Streams | Number of active media streams during an active period of time and over a lifetime period. |
| Participants | Total number of participants in session recordings during an active period of time and over a lifetime period. |

# Show rec redundancy

The **show rec redundancy** command displays information for session recording server statistics when the Oracle USM is configured for HA. These statistics include metadata monitored over an active period of time and over a lifetime period (where lifetime totals reflect from the last reboot of the Oracle USM to the present time) on both the primary and redundant Oracle USM. The following example shows the use of this command.

1. Log into the Oracle USM as a User or Superuser.

   ```
   ACMEPACKET> enable
   ACMEPACKET(enable)#
   ```

2. Type **show rec redundancy** and press Enter to display the session recording server statistics for Oracle USMs in HA mode. The following output is an example of the show rec redundancy command.

```
ACMEPACKET(enable)# show rec redundancy
```

Show rec redundancy output

Primary System

```
13:49:44-81645
Recording Agent Status        -- Period -- -------- Lifetime --------
                Active    High   Total      Total  PerMax    High
Rec Sessions         0       1       1          1       1       1
Comm Groups          0       0       0          0       0       0
Comm Sessions        0       1       1          1       1       1
Media Streams        0       2       2          2       2       2
Participants         0       2       2          2       2       2

Redundant System
13:49:44-81646
Recording Agent Status        -- Period -- -------- Lifetime --------
                Active    High   Total      Total  PerMax    High
Rec Sessions         0       1       1          1       1       1
Comm Groups          0       0       0          0       0       0
Comm Sessions        0       1       1          1       1       1
Media Streams        0       2       2          2       2       2
Participants         0       2       2          2       2       2
```

The following table describes the session recording server statistics in the **show rec redundancy** command output.

| Object | Description |
|---|---|
| Rec Sessions | Number of recording sessions during an active period of time and over a lifetime period. |
| Comm Groups | Number of active communication session recording groups during an active period of time and over a lifetime period. |
| Comm Sessions | Number of active communication sessions during an active period of time and over a lifetime period. |
| Media Streams | Number of active media streams during an active period of time and over a lifetime period. |
| Participants | Total number of participants in session recordings during an active period of time and over a lifetime period. |

# Inserting SIP Headers into SIPREC Metadata

The SIPREC Extension Data Enhancements SPL provides additional header information in the originating SIP messages metadata sent to the Interactive Session Recorder. With this SPL, you can introduce more options for recording policy decisions when using the SIPREC feature of the Oracle USM. The enhanced metadata also allows for the realm-id to be used as an indicator of the recording account. The SPL also provides configurable values that collect additional header information to store in the metadata.

When the SPL is configured, the SIPREC Extension Data Enhancements SPL is only triggered upon INVITE/UPDATE requests, and stores the additional header information in the metadata that is sent to the Net-Net Interactive Session Recorder (NN-ISR). Metadata is a XML MIME attachment that describes recording details to the Net-Net ISR.

By default, the **Extension-Headers** SPL option collects only the Request-URI in a received INVITE. You can store additional header information by configuring the SPL with additional attributes in the **spl-options** under the global **spl-config**.. The values must be in a comma separated list enclosed in double quotation marks. For example:

```
Extension-Headers="P-Asserted-Identity,Diversion"
```

This configuration of the **Extension-Headers** option adds the originating Request-URI along with all P-Asserted-Identity and Diversion-Headers into the participant section of the metadata.

You can configure the **LRE-Identifier** SPL option to add an identifier of the logical remote entity (LRE) that triggered the recording to the <apkt:realm> element of the extension metadata. When configured with a value added, the value appears in place of the identifier. When configured without a value, the identifier of the logical remote entity is used. For example, session-agent will be the hostname, realm-config will be the realm, and sip-interface will be the realm name.

> **Note:**
>
> Both options are required for the SPL to function properly.

## Sample Metadata

The sample below shows metadata with new extension data added by the SIPREC Extension Data Enhancements SPL (New metadata appears in bold):

```
<?xml version='1.0' encoding='UTF-8'?>
   <recording xmlns='urn:ietf:params:xml:ns:recording'>
      <datamode>complete</datamode>
         <session id="BYiC7uSZQGN3VQdzWI1HWw==">
            <associate-time>2012-06-26T13:44:13</associate-time>
         </session>
         <participant id="hq18GJs3TtJdhjPsfPNV8A=="
session="BYiC7uSZQGN3VQdzWI1HWw==">
            <nameID aor="sip:sipp@192.168.10.1">
               <name>sipp</name>
            </nameID>
            <send>aD50KX+LTvxNzASg+/GQTg==</send>
               <associate-time>2012-06-26T13:44:13</associate-time>
                  <extensiondata xmlns:apkt="http://acmepacket.com/siprec/
extensiondata">
                     <apkt:callingParty>true</apkt:callingParty>
                     <apkt:request-uri>sip:service@192.168.101.13:5060    </
apkt:request-uri>
                     <apkt:in-realm>net192</apkt:in-realm>
                     <apkt:header label=P-Asserted-Identity>
                        <value>sip:mike@acme.com</value>
                        <value>sip:bob@cisco.com</value></apkt:header>
                     <apkt:header label=Diversion>

<value>&lt;sip:jojo@divert.com&gt;;happy=days;green=envy</value>

<value>&lt;sip:bebe@MediaTen.net&gt;;green=monster;go=carts</value>
                           <value>&lt;tel:
+8675309;night=mare&gt;;gear=head;green=monitor</value></apkt:header>
                  </extensiondata>
                     </participant>
```

```
         <participant id="Ki6WEUi4TPRUPLtEaEhA7Q=="
session="BYiC7uSZQGN3VQdzWI1HWw==">
            <nameID aor="sip:service@192.168.101.13">
               <name>sut</name>
               </nameID>
               <send>f9NDVhyMTul+ePlM2SceQA==</send>
               <associate-time>2012-06-26T13:44:13</associate-time>
               <extensiondata xmlns:apkt="http://acmepacket.com/siprec/
extensiondata">
                  <apkt:callingParty>false</apkt:callingParty>
               </extensiondata>
         </participant>
         <stream id="aD50KX+LTvxNzASg+/GQTg=="session="BYiC7uSZQGN3VQdzWI1HWw==">
            <label>65804</label>
            <mode>separate</mode>
            <associate-time>2012-06-26T13:44:13</associate-time>
         </stream>
         <stream id="f9NDVhyMTul+ePlM2SceQA=="session="BYiC7uSZQGN3VQdzWI1HWw==">
            <label>65805</label>
            <mode>separate</mode>
            <associate-time>2012-06-26T13:44:13</associate-time>
        </stream>
   </recording>
```

## Configure SIP Headers for SIPREC Metadata

To get more detailed information about a recorded session, you can add more SIP headers within the SIPREC metadata by way of the **Extension-Headers** option. The default behavior stores only the Request-URI and realm-id.

You must configure the **Extension-Headers** option at the global level under **spl-config** because the session-agent, realm-config, and sip-interface configurations do not recognize the option. The first time you configure one or more extension headers, you need only to save and activate the configuration for the system to recognize the extension headers. When you modify the existing SPL extension header list you need to save and activate the configuration, and reboot the system for the changes to take effect. Real Time Configuration (RTC) does not apply to extension header options.

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

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

2. Type **spl-options +Extension-Headers="<value>"** , where <value> is the additional header information to store, and press Enter.

   ```
   ACMESYSTEM(spl-config)# spl-options +Extension-Headers="P-Asserted-
   Identity,Diversion"
   ```

3. Type **done** to save the configuration.

# SIPREC Recording Session Refresh

The Oracle USM provides for periodic in-dialog OPTIONS request/response exchanges to test the availability of the recording session dialog.

As shown below, and previously explained, establishment of a SIPREC session involves the creation of three distinct dialogs: a recording session dialog between the SIPREC client and

SIPREC server, a SIP dialog between the calling party (A) and the SBC (acting as a B2BUA), and another SIP dialog between the SBC (acting as a B2BUA) and the called party (B). The two SIP dialogs are viewed conceptually as a single communications session dialog.

Both the call recording and the communication session dialogs are triggered by an INVITE at the SIPREC client. Upon reception, the recording client (the SBC) buffers the original INVITE, and sends a copy if it, along with call-related meta-data to the recording server. A 200 OK response from the server establishes the recording session dialog that will carry replicated call content; signalling information is conveyed by the metadata in the recording dialog. After establishing the recording session, the client transmits the buffered INVITE to the called party to begin establishing the communications session dialog.

In the absence of a 200 OK response, the recording client is required to reject the offered INVITE with a 503 (Service Unavailable) SIP error code and an accompanying Reason header indicating that the recording session could not be established.

After establishing the recording session dialog, the client does not monitor its connection state. Its only opportunity to verify the state occurs after the termination of the communications session dialog, which triggers a client-originated BYE to the recording server. Reception of a 200 OK/ACK confirms the persistence of the recording session, while the lack of acknowledgement indicates the failure of the recording session at some unspecified point in time.

This failure to identify a prematurely terminated recording session has caused service providers and some regulatory agencies to require timely recognition of a failed call recording dialog.

Release S-CZ7.2.0, and later releases, address this requirement by providing for the exchange of periodic in-dialog OPTIONS request/response exchanges to test the availability of the recording session dialog.

## Timer_B

Section 17.1.1.2 of RFC 3261, SIP: Session Initiation Protocol describes a TIMER_B, the SIP INVITE transaction timer, which specifies the maximum interval between an INVITE request and response. The RFC suggests a default value of 32 seconds. Depending on the installed release version, two ACLI commands, `trans-expire` and `initial-inv-trans-expire`, provide user control over the TIMER_B value at both the SIP global level and the SIP interface level. Use these commands to adjust TIMER_B values if required.

## OPTIONS Request/Response

Availability of the recording session is tested by an in-dialog SIP OPTIONS request/response exchange initiated by the call recording client and completed by the call recording server. If the OPTIONS exchange is enabled, two timers are set when the recording session dialog is established: a refresh-timer that specifies the interval between OPTIONS requests sent by the call recording client, and a response-timer that specifies the maximum interval between the OPTIONS request and the OPTIONS response. The response-timer is set to the smaller of the configured TIMER_B or refresh-timer values.

Expiration of the refresh-timer results in the transmission of an OPTIONS request to the call recording server. The server, in turn, must reply with a 200 OK OPTIONS response prior to the expiration of the response-timer. In the event of a positive response, the call recording client restarts the refresh-timer, and re-issues the OPTIONS request when the timer next expires.

In the event that the call recording server response is non-positive, that is either

1. not received prior to the expiration of the response timer, or

2. a non-2xx OPTIONS response

the SBC terminates the session recording dialog and records the termination in the event log.

Typically, a SIPREC environment contains multiple recorders configured to record a communications session dialog, consequently the communications session should be terminated only when all recording dialogs in the call recording server have terminated. At that point the communications session dialog is torn down with a 503 (Service Unavailable) error report.

## Recording Session Refresh Configuration

Use the following procedure to enable a SIP OPTIONS request/response mechanism used to detect a prematurely terminated call recording session dialog.

1. Access the **session-recording-server** configuration element.

```
ORACLE# configure terminal
ORACLE(configure)# session-router
ORACLE(session-router)# session-recording-server
ORACLE(session-recording-server)#
```

2. Type **select** to choose and configure an existing object

```
ORACLE(session-recording-server)# select
<recordingServerName>:1: name=SRS02 desc=SRS Atlanta
2: name=name desc="SRS First Server:
3: name=srs01 desc=Miami

selection: 2
```

3. **session-recording-required**—ensure that this parameter is enabled.
   This is a prerequisite for the provision of recording session refresh services.
4. **refresh-interval**—use this parameter to both enable the SIP OPTIONS request/response mechanism, and to assign a value to the refresh-timer, which measures the maximum allowed interval (in seconds) between the OPTIONS request sent by the call-recording client and the OPTIONS response returned by the call-recording server.
   By default, refresh-interval is set to 0, which disables detection of a failed recording session dialog.
   Assignment of any non-zero value enables detection and sets the allowable interval between OPTIONS requests and responses.
   Consult local policy and regulatory requirements when specifying the refresh-interval value.

```
ACMEPACKET(session-router)# refresh-interval 12
ACMEPACKET(session-recording-server)#
```

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

# Codec Negotiation

In a SIPREC environment, it is assumed that the recording ecosystem provides transcoding media servers for which media calls can be redirected to, relieving the issue of codec matching from the recording servers. However, if transcoding media servers are not provided, the responsibility for transcoding falls on the recording server or the recording client in a SIPREC environment. The Oracle USM/SRC is required to impose some policy decisions on the codec negotiation between the three, or more, end-points. Specifically, the codec negotiation between the two participants and the recording server is subject to additional policy actions.

The SDP answer from the SRS may not agree with the media flows established in the communication session between UA-A and UA-B. If UA-A and UA-B agree to use G729, yet the SRS's answer indicates no support for G729, the SRS is then unable to interpret the media streams. The SDP offer forwarded to the called party (in this case UA-B) limits the codec choices to those supported by the SRS.

> **Note:**
>
> The recording agent forwards the original codec offer to the SRS prior to sending the invite to the UA-B. The SRS responds with the SDP answer, indicating the codec list most desirable to the SRS. The codec list in the answer is then forwarded to UA-B. This allows three parties in a conference call to participate in the negotiation of the codecs among the supported formats only.

# SIPREC Call Flows

This section provides examples of call flow scenarios that can occur in a SIPREC environment. SIP recording call flow examples include:

For Selective Recording:

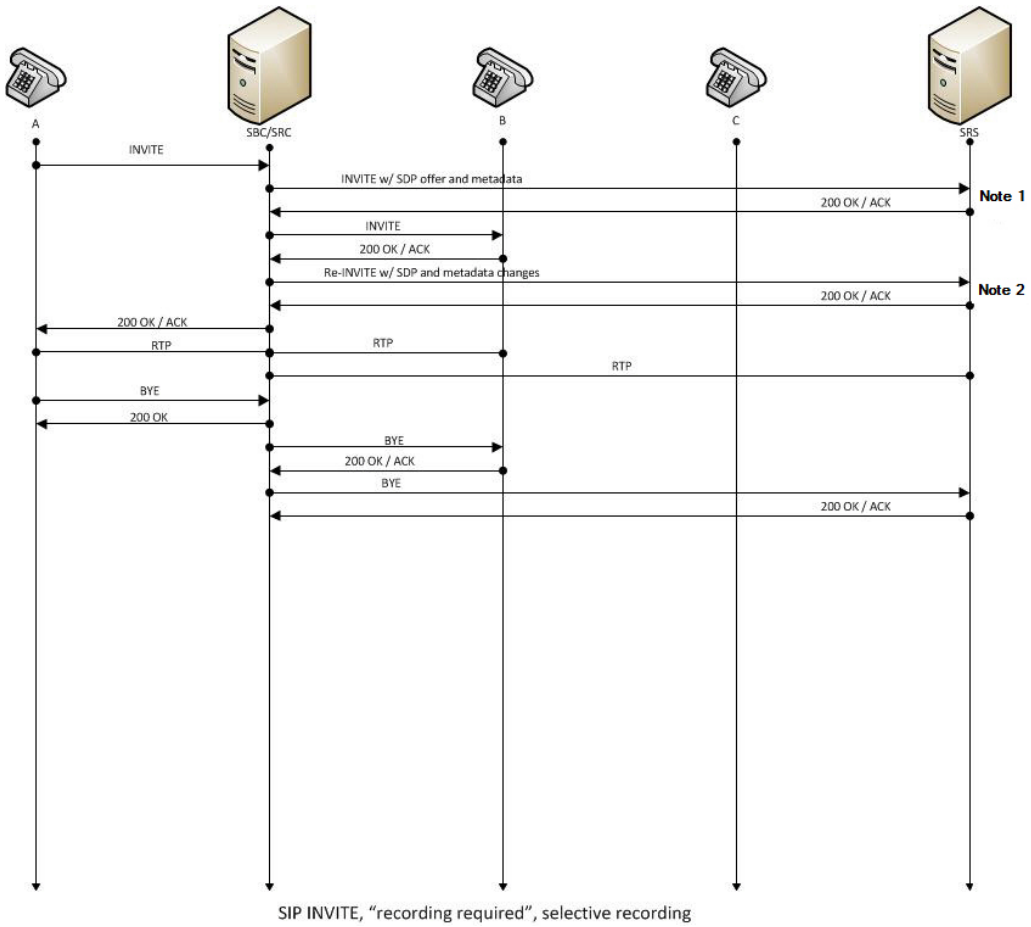
- Normal Call (recording required)
- Normal Call (recording not required)
- Early Media Call (recording not required)
- REFER Pass-Through Call (REFER handled by User Agent)
- REFER Call (REFER handled by the Oracle USM)
- SRS Indicates Busy in Call (recording not required)

> **Note:**
>
> REFER is a SIP method indicating that the recipient (identified by the Request-URI) should contact a third party using the contact information provided in the request.

## Selective Recording

### Normal Call (recording required)

The following illustration shows a normal call using selective recording with recording required. For SDP and Metadata information in Notes 1 and 2 , see Sample SDP and Metadata.




I

| Call Flow Description | |
|---|---|
| ① UA-A sends INVITE to Oracle USM. | ⑩ RTP stream initiated between Oracle USM and UA-B. |
| ② Oracle USM forwards INVITE with SDP and metadata to SRS. | ⑪ RTP stream initiated between Oracle USM and SRS. |
| ③ SRS responds with OK to Oracle USM. | ⑫ UA-A sends BYE to Oracle USM. |
| ④ Oracle USM sends INVITE to UA-B. | ⑬ Oracle USM responds with OK to UA-A. |
| ⑤ UA-B responds with OK to Oracle USM. | ⑭ Oracle USM sends BYE to Oracle USM. |
| ⑥ Oracle USM sends re-INVITE with SDP and metadata changes to SRS. | ⑮ Oracle USM responds with OK to UA-A. |
| ⑦ SRS responds with OK to Oracle USM. | ⑯ Oracle USM sends BYE to UA-B. |
| ⑧ Oracle USM forwards OK response to UA-A. | ⑰ UA-B responds with OK to Oracle USM. |
| ⑨ RTP stream initiated between UA-A and Oracle USM. | ⑱ Oracle USM sends BYE to SRS. |
| | ⑲ SRS responds with OK to Oracle USM. |

# Sample SDP and Metadata

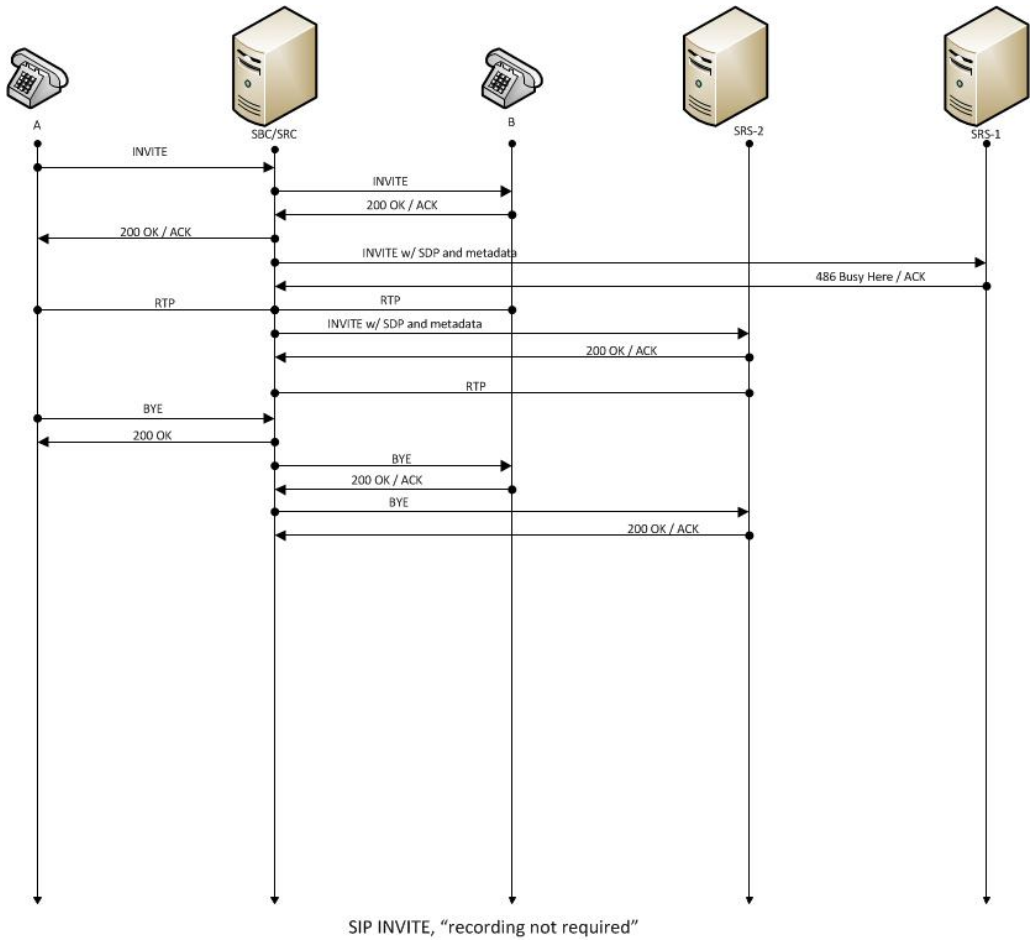
The following sample SDP and Metadata pertain to Notes 1 and 2 in the previous Call Flow diagram.

```
--[Note 1]-----------------------------
Content-Type: application/sdp
v=0
o=- 171 213 IN IP4 10.0.0.2
s=-
c=IN IP4 10.0.0.1
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=label:1

Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session
<?xml version='1.0' encoding='UTF-8'?>
<recording xmlns='urn:ietf:params:xml:ns:recording'>
        <dataMode>complete</dataMode>
        <session id="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <start-time>2011-06-27T17:03:57</start-time>
        </session>
        <participant id="urn:uuid:10ac9063-76b7-40bb-4587-08ba290d7327"
session="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <aor>sip:sipp@168.192.24.40</aor>
                <name>sipp </name>
                <send>urn:uuid:07868c77-ef8e-4d6f-6dd5-a02ff53a1329</send>
                <start-time>2011-06-27T17:03:57</start-time>
        </participant>
        <participant id="urn:uuid:797c45f5-e765-4b12-52b0-d9be31138529"
session="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <aor>sip:service@168.192.24.60</aor>
                <name>sut </name>
        </participant>
        <stream id="urn:uuid:4a72a1ed-abb2-4d7c-5f4d-6d4c36e2d4ec"
session="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <mode>separate</mode>
                <start-time>2011-06-27T17:03:57</start-time>
                <label>1</label>
        </stream>
</recording>


--[Note 2]-----------------------------
Content-Type: application/sdp
v=0
o=- 171 213 IN IP4 10.0.0.2
s=-
c=IN IP4 10.0.0.1
t=0 0
m=audio 6000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=label:1
m=audio 6002 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=label:2

Content-Type: application/rs-metadata+xml
Content-Disposition: recording-session
<?xml version='1.0' encoding='UTF-8'?>
<recording xmlns='urn:ietf:params:xml:ns:recording'>
        <dataMode>partial</dataMode>
        <session id="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <start-time>2011-06-27T17:03:57</start-time>
```

```
        </session>
        <participant id="urn:uuid:797c45f5-e765-4b12-52b0-d9be31138529"
session="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <aor>sip:service@168.192.24.60</aor>
                <name>sut </name>
                <send>urn:uuid:4a72a1ed-abb2-4d7c-5f4d-6d4c36e2d4ec</send>
                <start-time>2011-06-27T17:03:58</start-time>
        </participant>
        <stream id="urn:uuid:07868c77-ef8e-4d6f-6dd5-a02ff53a1329"
session="urn:uuid:79b2fcd8-5c7f-455c-783f-db334e5d57d0">
                <mode>separate</mode>
                <start-time>2011-06-27T17:03:58</start-time>
                <label>2</label>
        </stream>
</recording>
```

# Normal Call (recording not required)

The following illustration shows a normal call using selective recording with recording optional.

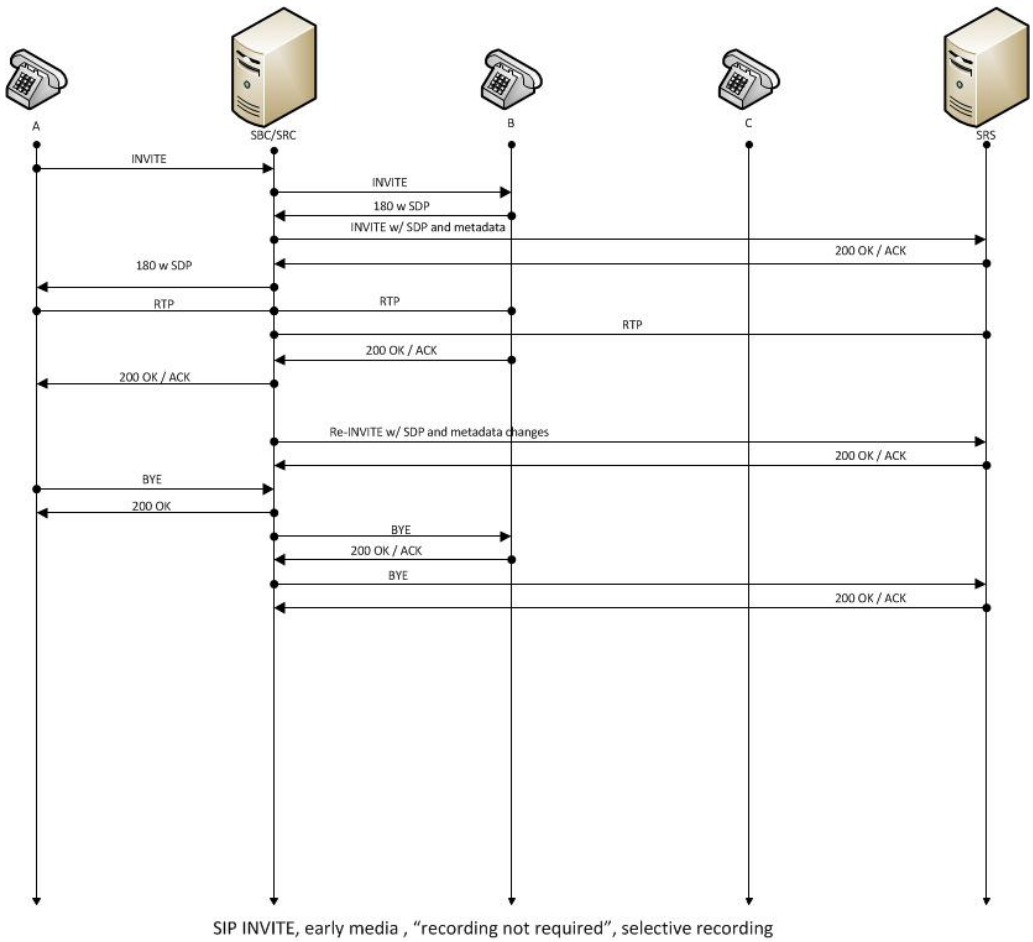



SIP INVITE, “recording not required”

| Call Flow Description | |
|---|---|
| ① UA-A sends INVITE to Oracle USM. | ⑧ RTP stream initiated between Oracle USM and SRS. |
| ② Oracle USM forwards INVITE to UA-B. | ⑨ UA-A sends BYE to Oracle USM. |
| ③ UA-B responds with OK to Oracle USM. | ⑩ Oracle USM responds with OK to UA-A. |
| ④ Oracle USM forwards OK response to UA-A. | ⑪ Oracle USM sends BYE to UA-B. |
| ⑤ Oracle USM sends INVITE with SDP and metadata to SRS. | ⑫ UA-B responds with OK to Oracle USM. |
| ⑥ SRS responds with OK to Oracle USM. | ⑬ Oracle USM sends BYE to SRS. |
| ⑦ RTP stream initiated between UA-A, Oracle USM, and UA-B. | ⑭ SRS responds with OK to Oracle USM. |

# Early Media Call (recording not required)

The following illustration shows an early media call using selective recording with recording optional.

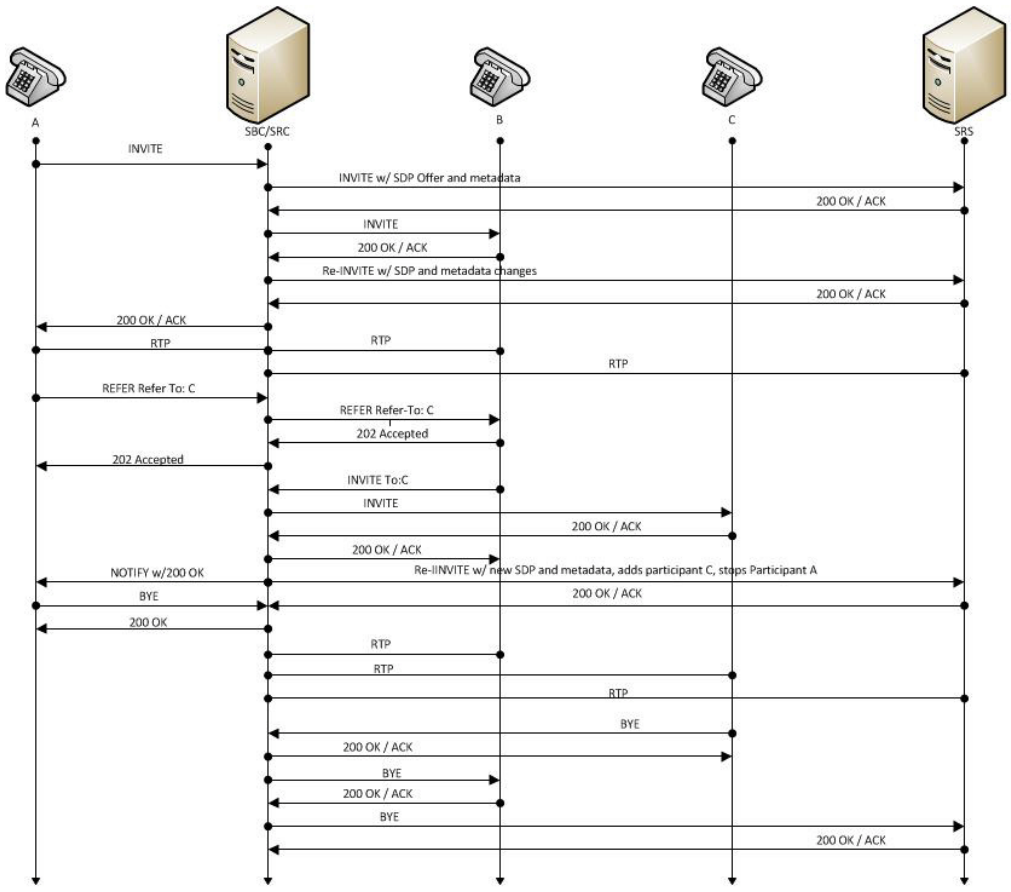



SIP INVITE, early media , “recording not required”, selective recording

| Call Flow Description | |
|---|---|
| ① UA-A sends INVITE to Oracle USM. | ⑩ UA-B responds with OK to Oracle USM. |
| ② Oracle USM forwards INVITE to UA-B. | ⑪ Oracle USM forwards OK to UA-A. |
| ③ UA-B sends 180 and SDP to Oracle USM. | ⑫ Oracle USM sends re-INVITE with SDP and metadata changes to SRS. |
| ④ Oracle USM sends INVITE with SDP and metadata to SRS. | ⑬ SRS responds with OK to Oracle USM. |
| ⑤ SRS responds with OK to Oracle USM. | ⑭ UA-A sends BYE to Oracle USM. |
| ⑥ Oracle USM sends 180 with SDP to UA-A. | ⑮ Oracle USM responds with OK to UA-A. |
| ⑦ RTP stream initiated between Oracle USM and UA-A. | ⑯ Oracle USM sends BYE to UA-B. |
| ⑧ RTP stream initiated between Oracle USM and UA-B. | ⑰ UA-B responds with OK to Oracle USM. |
| ⑨ RTP stream initiated between Oracle USM and SRS. | ⑱ Oracle USM sends BYE to SRS. |
| | ⑲ SRS responds with OK to Oracle USM. |

# REFER Pass-Through Call (REFER handled by User Agent)

The following illustration shows a REFER pass-through call using selective recording and the User Agent (UA) handling the REFER on the call. Recording is required in this call flow.

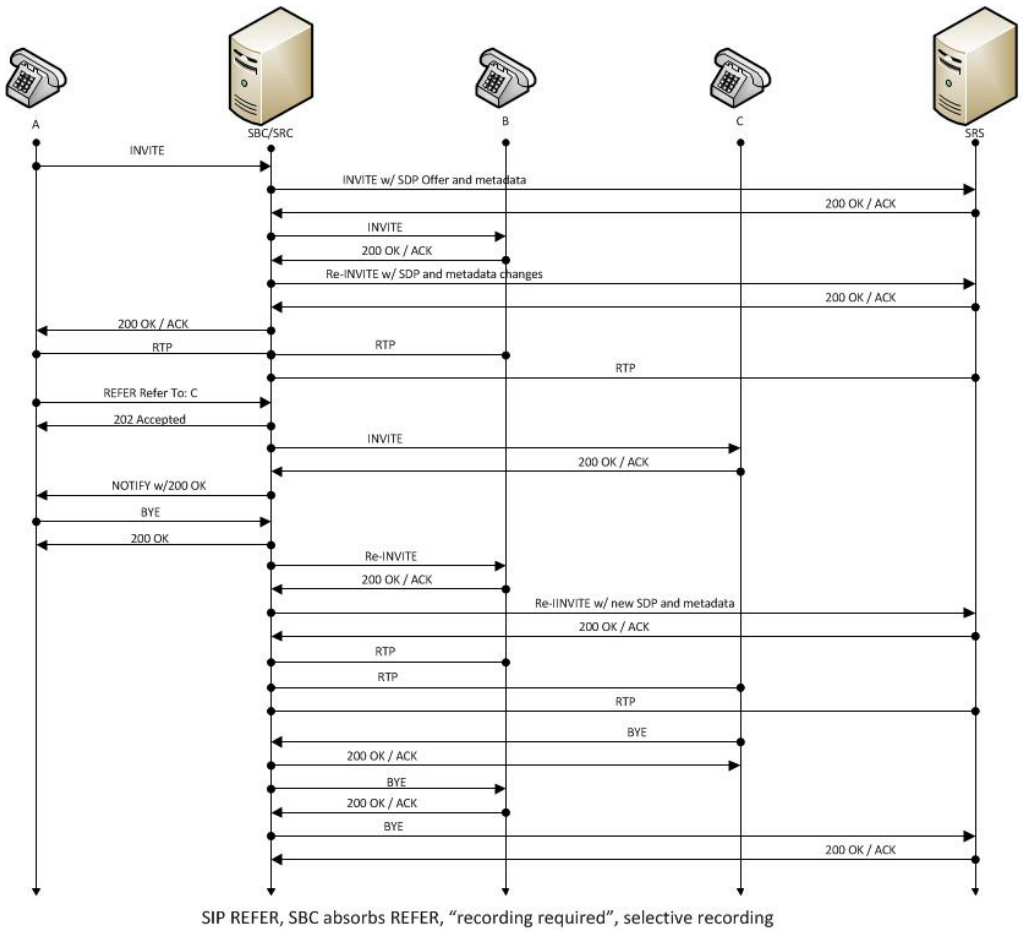



SIP REFER, UA handles REFER, "recording required", selective recording

| Call Flow Description | |
|---|---|
| 1 - UA-A sends INVITE to Oracle USM. | 18 - UA-C responds with OK to Oracle USM. |
| 2 - Oracle USM forwards INVITE with SDP Offer and metadata to SRS. | 19 - Oracle USM forwards OK response to UA-B. |
| 3 - SRS responds with OK to Oracle USM. | 20 - Oracle USM sends NOTIFY with OK reponse to UA-A. |
| 4 - Oracle USM sends INVITE to UA-B. | 21 - Oracle USM sends re-INVITE to SRS with new SDP and metadata, adds participant C, stops participant A . |
| 5 - UA-B responds with OK to Oracle USM. | 22 - SRS responds with OK to Oracle USM. |
| 6 - Oracle USM sends re-INVITE with SDP and metadata changes to SRS. | 23 - UA-A sends BYE to Oracle USM. |
| 7 - SRS responds with OK to Oracle USM. | 24 - Oracle USM responds with OK to UA-A. |
| 8 - Oracle USM forwards OK response to UA-A. | 25 - Oracle USM responds with OK to UA-A. |
| 9 - RTP stream initiated between UA-A and Oracle USM. | 26 - RTP stream initiated between Oracle USM and UA-B. |
| 10 - RTP stream initiated between Oracle USM and UA-B. | 27 - RTP stream initiated between Oracle USM and UA-C. |
| 11 - RTP stream initiated between Oracle USM and SRS. | 28 - RTP stream initiated between Oracle USM and SRS. |
| 12 - UA-A sends REFER-TO: C to Oracle USM. | 29 - UA-C sends BYE to Oracle USM. |
| 13 - Oracle USM forwards REFER-TO: C to UA-B. | 30 - Oracle USM responds with OK to UA-C. |
| 14 - UA-B responds with 202 ACCEPTED to Oracle USM. | 31 - Oracle USM sends BYE to UA-B. |
| 15 - Oracle USM forwards 202 ACCEPTED to UA-A. | 32 - UA-B responds with OK to Oracle USM. |
| 16 - UA-B sends INVITE TO: C to Oracle USM. | 33 - Oracle USM sends BYE to SRS |
| 17 - Oracle USM sends INVITE to UA-C. | 34 - SRS responds with OK to Oracle USM. |

# REFER Call (REFER handled by Oracle USM)

The following illustration shows a call using selective recording and the Session Border Controller (Oracle USM) handling the REFER on the call. Recording is required in this call flow.



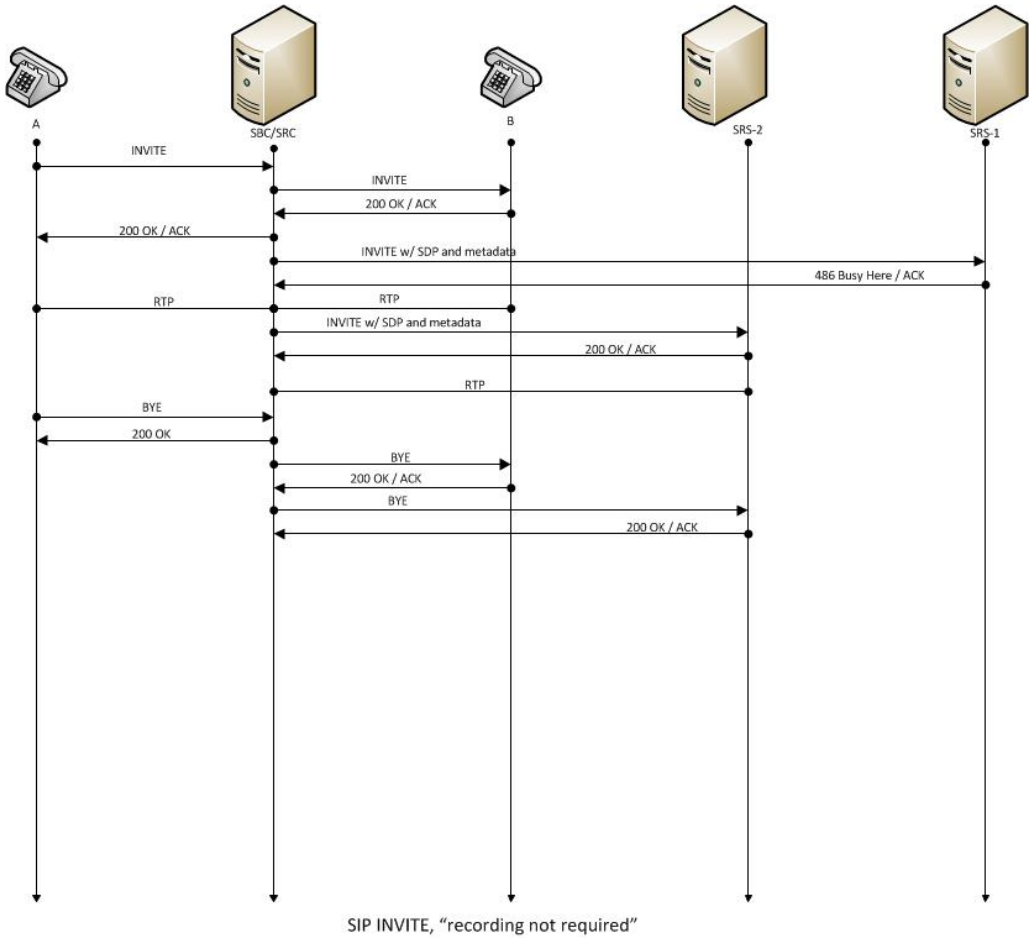

SIP REFER, SBC absorbs REFER, "recording required", selective recording

| Call Flow Description | |
|---|---|
| 1 - UA-A sends INVITE to Oracle USM. | 16 - Oracle USM sends NOTIFY with OK response to UA-A. |
| 2 - Oracle USM forwards INVITE with SDP Offer and metadata to SRS. | 17 - UA-A sends BYE to Oracle USM. |
| 3 - SRS responds with OK to Oracle USM. | 18 - Oracle USM responds with OK to UA-A. |
| 4 - Oracle USM sends INVITE to UA-B. | 19 - Oracle USM sends re-INVITE to UA-B. |
| 5 - UA-B responds with OK to Oracle USM. | 20 - UA-B responds with OK to Oracle USM. |
| 6 - Oracle USM sends re-INVITE with SDP and metadata changes to SRS. | 21 - Oracle USM sends re-INVITE to SRS with new SDP and metadata. |
| 7 - SRS responds with OK to Oracle USM. | 22 - SRS responds with OK to Oracle USM. |
| 8 - Oracle USM forwards OK response to UA-A. | 23 - RTP stream initiated between Oracle USM and UA-B. |
| 9 - RTP stream initiated between UA-A and Oracle USM. | 24 - RTP stream initiated between Oracle USM and UA-C. |
| 10 - RTP stream initiated between Oracle USM and UA-B. | 25 - RTP stream initiated between Oracle USM and SRS. |
| 11 - RTP stream initiated between Oracle USM and SRS. | 26 - UA-C sends BYE to Oracle USM. |
| 12 - UA-A sends REFER-TO: C to Oracle USM. | 27 - Oracle USM responds with OK to UA-C. |
| 13 - Oracle USMOracle USM responds with 202 ACCEPTED to UA-A. | 28 - Oracle USM sends BYE to UA-B. |

| Call Flow Description | |
|---|---|
| 14 - Oracle USM sends INVITE to UA-C. | 29 - UA-B responds with OK to Oracle USM. |
| 15 - UA-C responds with OK to Oracle USM. | 30 - Oracle USM sends BYE to SRS. |
| | 31 - SRS responds with OK to Oracle USM. |

# SRS Indicates Busy in Call (recording not required)

The following illustration shows the Session Recording Server (SRS) is BUSY for a call session. Recording is not required in this call flow.

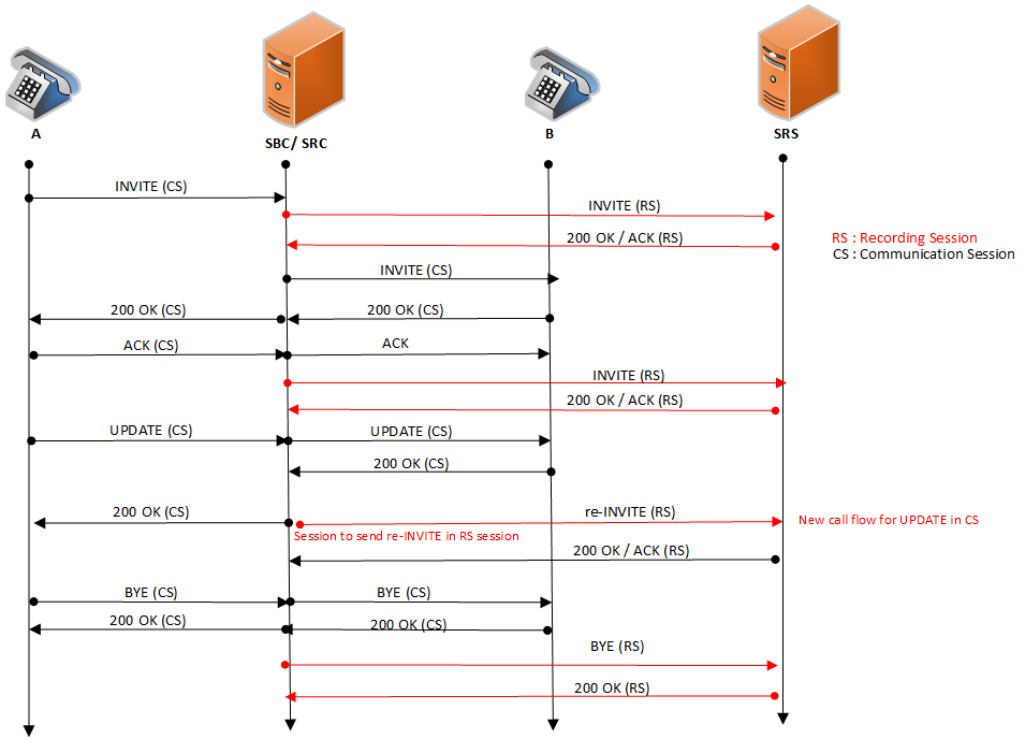



| Call Flow Description | |
|---|---|
| ① UA-A sends INVITE to Oracle USM. | ⑨ Oracle USM sends INVITE to SRS2 with SDP and metadata. |
| ② Oracle USM forwards INVITE to UA-B. | ⑩ SRS2 responds with OK to Oracle USM. |
| ③ UA-B responds with OK to Oracle USM. | ⑪ RTP stream initiated between Oracle USM and SRS2. |
| ④ Oracle USM forwards OK response to UA-A. | ⑫ UA-A sends BYE to Oracle USM. |

| Call Flow Description | |
|---|---|
| ⑤ Oracle USM sends INVITE to SRS1 with SDP and metadata. | ⑬ Oracle USM responds with OK to UA-A. |
| ⑥ SRS1 responds to Oracle USM with 436 BUSY HERE. | ⑭ Oracle USM sends BYE to UA-B. |
| ⑦ RTP stream initiated between UA-A andOracle USM. | ⑮ UA-B responds with OK to Oracle USM. |
| ⑧ RTP stream initiated between Oracle USM and UA-B. | ⑯ Oracle USM sends BYE to SRS2. |
| | ⑰ SRS2 responds with OK to Oracle USM. |

# Call Transfer Scenario (recording required)

The following illustration shows the Re-INVITE from Oracle USM to SRS on receiving in-dialog-requests INVITE/ UPDATE/ Re-INVITE during call session. Recording is required in this call flow.




Call Flow (when **session-recording-server** is enabled)

| Call Flow Description | |
|---|---|
| 1 - UA-A sends INVITE to Oracle USM. | 11 - UA-B responds with OK to Oracle USM. |
| 2 - Oracle USM updates the INVITE message to the Metadata in SRS. | 12 - Oracle USM forwards OK to UA-A.. |
| 3 - SRS responds with OK to Oracle USM. | 13 - Oracle USM sends a re-INVITE to SRS in Recording Session. |
| 4 - Oracle USM sends an INVITE to UA-B. | 14 - SRS responds with OK to Oracle USM. |

| Call Flow Description | |
|---|---|
| 5 - UA-B responds with OK to Oracle USM. | 15 - UA-A sends BYE to Oracle USM. |
| 6 - Oracle USM forwards OK response to UA-A. | 16 - Oracle USM forwards BYE to UA-B. |
| 7 - Oracle USM sends INVITE to SRS in Recording Session. | 17 - UA-B responds with OK to Oracle USM. |
| 8 - SRS responds with OK to Oracle USM. | 18 - Oracle USM forwards OK to UA-B. |
| 9 - UA-A sends an UPDATE message to Oracle USM. | 19 - Oracle USM sends BYE to SRS in Recording Session. |
| 10 - Oracle USM forwards UPDATE message to UA-B. | 20 - SRS responds with OK to Oracle USM in Recording Session. |

# Oracle Communications Session Monitor Mediation Engine

The Oracle® Communications Session Monitor Mediation Engine is a platform that collects SIP, DIAMETER, DNS and ENUM protocol message traffic received from Communications Session Monitor Probes. A Probe is software run on COTS hardware; it is deployed within a network and collects packets from span/monitor ports on Ethernet switches, or receives IP-in-IP tunneled packet-traces from Oracle's Oracle USM. A Probe takes the protocol packets, prepends a receive timestamp and other information, encapsulates the packets, and passes them to the Communications Session Monitor via a secure connection. After receiving protocol traffic from a Probe, the Communications Session Monitor stores the traffic in an internal database, and analyzes aggregated data to provide comprehensive multi-level monitoring, troubleshooting, and interoperability information.

In contrast to the Packet-Trace feature, message logging is performed by software, which sends a copy of sent/received messages over UDP, or by saving such messages in a local file. The copy includes a timestamp, port/vlan information, and IP:port information, but all in ASCII format. Message Logging is performed after all decryption, meaning that SIP/TLS traffic cam be monitored. Because remote message logging sends the protocol messages over UDP, there is no guarantee or confirmation of delivery.

The Oracle USM provides support for Communications Monitoring, a user-configurable capability that enables the system to function as a Communications Session Monitor Probe. Acting as a Probe, or as an exporter, the Oracle USM can:

1. Establish an authenticated, persistent, reliable TCP connection between itself and one or more Communications Session Monitor Mediation Engines.
2. Optionally ensure message privacy by encrypting the TCP connection using TLS.
3. Use the TCP connection to send a UTC-timestamped, unencrypted copy of a protocol message to the Communications Session Monitor Engine(s).
4. Accompany the copied message with related data to include: the port/vlan on which the message was sent/received, local and remote IP:port information, and the transport layer protocol.

# IPFIX

The Oracle USM uses the IPFIX suite of standards to export protocol message traffic and related data to the Communications Session Monitor Mediation Engine.

- RFC 5101, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*
- RFC 5102, *Information Model for IP Flow Information Export*
- RFC 5470, *Architecture for IP Flow Information Export*
- RFC 5655, *Specification of the IP Flow Information Export (IPFIX) File Format*
- RFC 5815, *Definitions of Managed Objects for IP Flow Information Export*

The IPFIX standards describe the use of templates to format the export of specific types of protocol traffic. The Oracle USM and the Communications Session Monitor Mediation Engine share ten (10) pre-defined templates that facilitate protocol message exchange, and subsequent processing and analysis by the Communications Session Monitor Engine.

The pre-defined templates are:

- incoming SIP/DNS over UDP
- incoming SIP over TCP
- incoming SIP over SCTP
- incoming DNS over UDP (entire IP and UDP header not included)
- outgoing SIP/DNS over UDP
- outgoing SIP over TCP
- outgoing SIP over SCTP
- outgoing DNS over UDP (entire IP and UDP header not included)
- media qos and flow record
- IPFIX handshake (used for connection establishment)

# Oracle Communications Operations Monitor Configuration

Oracle USM (USM) configuration on the Oracle Communications Operation Monitor (OCOM) consists of the following steps.

> **Note:**
>
> Enabling the comm-monitor so that the USM sends call information to OCOM results in a significant performance load. Contact Oracle Customer Support for more information.

1. Configure one or more USM-OCOM exporter-collector pairs. See "Configure the Oracle Communications Operations Monitor."
2. Optional—Assign a TLS profile to an exporter-collector pair. See "TLS Profile Configuration."

# Configure the Oracle Communications Operations Monitor

Use the following procedure to configure the Oracle Communications Operations Monitor (OCOM).

> **Note:**
>
> Enabling the comm-monitor so that the Oracle USM (USM) sends call information to OCOM results in a significant performance load. Contact Oracle Customer Support for more information.

From superuser mode, access comm-monitor configuration mode. Establish a connection between the USM, acting as a exporter of protocol message traffic and related data, and an OCOM Mediation Engine, acting as an information collector.

1. Access the comm-monitor configuration.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# system
   ACMEPACKET(system)# system-config
   ACMEPACKET(system-config)# comm-monitor
   ACMEPACKET(comm-monitor)#
   ```

2. **state**—Enable or disable communication monitoring. Default: disabled.

   ```
   ACMEPACKET(comm-monitor)# state enabled
   ACMEPACKET(comm-monitor)#
   ```

3. **sbc-group-id**—Retain the default or assign another integer value to the USM in its role as an information exporter. Default: 0.

   ```
   ACMEPACKET(comm-monitor)# sbc-group-id 5
   ACMEPACKET(comm-monitor)#
   ```

4. **qos-enable** —Enable or disable the export of RTP, SRTP, and QOS data flow information.

   ```
   ACMEPACKET(comm-monitor)# qos-enable enabled
   ACMEPACKET(comm-monitor)#
   ```

5. **interim-qos-update**—Enable or disable 10 second interim QoS update.

   ```
   ACMEPACKET(comm-monitor)# interim-qos-enable enabled
   ACMEPACKET(comm-monitor)#
   ```

6. **monitor-collector** —Move to monitor-collector configuration mode.

   In this mode you identify an OCOM Mediation Engine collector by IP address and port number.

   ```
   ACMEPACKET(comm-monitor)# monitor-collector
   ACMEPACKET(monitor-collector)#
   ```

7. **address** and **port**—Specify the IP address and port number monitored by an OCOM Mediation Engine for incoming IPFIX traffic.

   Enter an IPv4 address and a port number with values either 4739 (unsecured) or 4740 (secured). Default: 4739.

```
ACMEPACKET(monitor-collector)# address 172.30.101.239
ACMEPACKET(monitor-collector)# port 4739
ACMEPACKET(monitor-collector)#
```

8. Use the **network-interface** parameter to specify the network interface that supports the TCP connection between the USM to the OCOM Mediation Engine.

   - To specify the wancom0 management interface:

     ```
     ACMEPACKET(comm-monitor)# network-interface wancom0:0
     ACMEPACKET(comm-monitor)#
     ```

   - To specify a media interface:

     ```
     ACMEPACKET(comm-monitor)# network-interface m01
     ACMEPACKET(comm-monitor)#
     ```

9. Type**done** and **exit** to return to comm-monitor configuration mode.
10. Type **done**, **exit**, and **verify-config** to complete configuration.
11. Repeat the procedure to configure additional connections to OCOM, as needed.

- Optional—If the network interface specified in Step 8 is a media interface, you can optionally use TLS to encrypt the exporter-collector connection. To enable TLS encryption, use the **tls-profile** parameter to identify a TLS profile to be assigned to the network interface. The absence of an assigned TLS profile (the default state) results in unencrypted transmission. See TLS Profile Configuration for configuration details.

  ```
  ACMEPACKET(comm-monitor)# tls-profile commMonitor
  ACMEPACKET(comm-monitor)#
  ```

# TSCF Rekey Profile Configuration

Rekeying is a cryptographic technique that enhances security by enforcing the negotiation of existing keys on an ongoing secure connection. Rekeying can be either time-based, in which case new keys are negotiated at the expiration of a timer, or traffic-based, in which case new keys are negotiated when a threshold byte count is exceeded.

Use the following procedure to configure an optional tscf-rekey-profile. Later, you will assign the profile to a specific TSCF interface. If you do not intend to enforce re-keying, this procedure can be safely ignored.

1. From superuser mode, use the following command sequence to access tscf-rekey-profile configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tscf
   ACMEPACKET(tscf)# tscf-rekey-profile
   ACMEPACKET(tscf-rekey-profile)#
   ```

2. Use the **name** parameter to provide a unique identifier for this tscf-rekey-profile.

   ```
   ACMEPACKET(tscf-rekey-profile)# name tscfRekey01
   ACMEPACKET(tscf-rekey-profile)#
   ```

3. Use the **initiator** parameter to identify the rekey initiator.

   Supported values are **client** (default) | **server** (the Session Director)

   ```
   ACMEPACKET(tscf-rekey-profile)# initiator client
   ACMEPACKET(tscf-rekey-profile)#
   ```

4. Use the **max-rekey-time** parameter to specify the maximum interval (in minutes) between re-keying operations.

   Supported values are **0** (default) | **30 - 1440** (minutes)

   The default value, **0**, specifies that time-based rekeying is not enforced; other integer values specify that time-based re-keying must be initiated by the tunnel endpoint designated by the **initiator** parameter.

   ```
   ACMEPACKET(tscf-rekey-profile)# max-rekey-time 30
   ACMEPACKET(tscf-rekey-profile)#
   ```

5. Use the **max-rekey-data** parameter to specify the maximum traffic exchange (measured in Kb) between rekeying operations.

   The default value, **0**, specifies that traffic-based rekeying is not enforced; other integer values specify that traffic-based re-keying must be initiated by the tunnel endpoint designated by the **initiator** parameter.

   ```
   ACMEPACKET(tscf-rekey-profile)# max-rekey-data 0
   ACMEPACKET(tscf-rekey-profile)#
   ```

6. Use **done**, **exit**, and **verify-config** to complete tscf-rekey-profile configuration.
7. Repeat Steps 1 through 6 to configure additional tscf-rekey-profiles as required.

# TLS Profile Configuration

Use the following procedure to configure a tls-profile that identifies the cryptographic resources, specifically certificates and protocols, required for the establishment of a secure/encrypted connection between the Oracle USM and the Communications Session Monitor Mediation Engine.

1. From superuser mode, use the following command sequence to access tls-profile configuration mode.

   ```
   ACMEPACKET# configure terminal
   ACMEPACKET(configure)# security
   ACMEPACKET(security)# tls-profile
   ACMEPACKET(tls-profile)#
   ```

2. Use the **name** parameter to provide a unique identifier for this tls-profile.

   ```
   ACMEPACKET(tls-profile)# name commMonitor
   ACMEPACKET(tls-profile)#
   ```

3. Use the required **end-entity-certificate** parameter to specify the name of the certificate-record configuration that identifies the credential (specifically, an X509.v3 certificate) offered by the Oracle USM in support of its asserted identity.

   ```
   ACMEPACKET(tls-profile)# end-entity-certificate commMonitor509
   ACMEPACKET(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 the offered certificate. These referenced certificates are conveyed to the Communications Session Monitor Mediation Engine as part of the TLS exchange.

   Provide a comma separated list of existing CA **certificate-record** configuration elements.

   ```
   ACMEPACKET(tls-profile)# trusted-ca-certificates verisignClass3-
   a,verisignClass3-b,baltimore,thawtePremium,acme-CA
   ACMEPACKET(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 end-entity certificate received from the Communications Session Monitor Mediation Engine.

   Provide an integer within the range 1 through 10 (the default).

   The Oracle USM supports the processing of certificate chains (consisting of an end-entity certificate and some number of CA certificates) when X.509v3 certificate-based authentication is used. The following process validates a received TLS certificate chain.

   - 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
   - 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.
   - 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.
   - 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.
   - Check the X509v3 Extensions field to verify that the current certificate identifies a CA. If not so, authentication fails; otherwise, continue chain validation.
   - 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).
   - Compute the same MD5 hash. If the results are not identical, authentication fails; otherwise, continue chain validation.
   - 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.

   ```
   ACMEPACKET(tls-profile)# verify-depth 8
   ACMEPACKET(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.

   ```
   ACMEPACKET(tls-profile)# mutual-authenticate disabled
   ACMEPACKET(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.

# Anonymize Sensitive Data in SIP Messages

When you allow people to examine SIP INVITE messages in the Oracle Communications Operations Monitor (OCOM), you might want to hide certain sensitive information from their view for security and confidentiality reasons. For example, you might want to hide the SUBJECT header in the message and in the CPIM body, as well as the MIME content of the CPIM body. Oracle's solution is to provide an option to anonymize such information for display in OCOM.

When you enable the **anonymize-invite** option, the system makes a copy of the inbound SIP INVITE and allows the original to continue on its way. In the copy, the system parses the body of the INVITE and replaces the SUBJECT header and MIME content with a hyphen (-). No other message content is affected, and the full functionality of the OCOM remains available. When the troubleshooter views the SIP INVITE message, OCOM displays the anonymized copy of the SIP INVITE.

The default setting for the **anonymize-invite** option is disabled. Use the options parameter in the comm-monitor configuration to enable anonymize-invite.

> **Note:**
>
> Enabling the **anonymize-invite** option adds an additional CPU load to the Oracle USM (USM). Contact Oracle Customer Support for more information.

# Enable Anonymization in a SIP INVITE Message

When you want to hide certain sensitive information in a SIP INVITE message that the Oracle Communications Operations Monitor (OCOM) can display, you can configure the Oracle USM (USM) to anonymize the SUBJECT header in the message and in the CPIM body, as well as the MIME content of the CPIM body with the **anonymize-invite** option.
The default setting for the anonymize-invite option is disabled. Use the options parameter in the comm-monitor configuration to enable anonymize-invite.

> **Note:**
>
> Enabling the **anonymize-invite** option adds an additional CPU load to the USM. Contact Oracle Customer Support for more information.

1. Access the **comm-monitor** configuration element.

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

2. Select the comm-monitor instance that you want to enable for anonymization.
3. Go to the **options** parameter, type **anonymize-invite**, and press ENTER.
4. Save and exit the configuration.

# Oracle Communications Session Monitor Statistics

The Oracle USM collects statistics on the operations of its communications monitor probe, which provides protocol traffic information to Oracle Communications' Session Monitor. The user displays information about the connections to Session Monitor servers using the **show comm-monitor** command. The user can set all comm-monitor statistics to zero using the command **reset comm-monitor**.

This Command shows three types of aggregate statistics for all clients, including:

- Client State
- Socket Statistics
- Other Aggregate Statistics

The Oracle USM's **show comm-monitor** command shows the Lifetime statistics. The command runs without or with arguments, including:

- **show comm-monitor**—Shows client connection states and aggregate stats
- **show comm-monitor by-client <IP-Addr>**—Shows client stats
- **show comm-monitor errors**—Shows Errors captured by the Session Monitor
- **show comm-monitor internal**—Shows Oracle USM traffic-specific counters

Example command output without arguments is presented below:

```
ORACLE# show comm-monitor
Client                    State         PROTOCOL
================================================
192.168.42.10:4739      Out-of-Service       TCP
192.168.42.11:4739      Connecting           TCP
192.168.42.12:4739      In-Service           TCP

15:06:10-35 (recent)
CommMonitor Socket Statistics                    --- Lifetime ---
                                       Recent      Total     PerMax
=====================================================================
Socket Message Dropped                    0          0          0
Socket Send Error                         0          0          0
Socket Not Ready                          0          0          0
Socket Timeouts                           0          0          0
Socket Disconnects                        0          0          0
Socket Reconnects                         0          0          0
Buffer Allocation Error                   0          0          0

CommMonitor Statistics                           --- Lifetime ---
                                       Recent      Total     PerMax
=====================================================================
Handshake Msg Sent                        0          0          0
Handshake Msg ACK                         0          0          0
Handshake Msg NAK                         0          0          0
Keep Alive                                0          0          0
SIP UDP Recv Msg Sent                     0          0          0
SIP UDP Send Msg Sent                     0          0          0
SIP TCP Recv Msg Sent                     0          0          0
SIP TCP Send Msg Sent                     0          0          0
SIP SCTP Recv Msg Sent                    0          0          0
SIP SCTP Send Msg Sent                    0          0          0
```

```
ENUM Recv Msg Sent                    0          0          0
ENUM Send Msg Sent                    0          0          0
```

# Packet Trace

The Oracle USM's packet trace tool provides the user with the ability to capture traffic from the Oracle USM itself.

The user invokes the tool from the ACLI, manually specifying:

- How to capture (local vs remote)
- What to capture
- Capture start and stop

There are two capture modes, one that saves traffic locally and one that mirrors traffic to a user-specified target. Software only deployments support local capture only. Proprietary Acme hardware deployments support both local and remote capture.

- Local capture supports PCAP filters to specify the type of traffic to capture. Remote capture supports its own syntax to identify the traffic to mirror.
- Local packet capture is dependent on access control configuration, not capturing any denied traffic. Remote capture mirrors traffic regardless of access control configuration.
- The system does not capture RTP via local packet capture.

Installed NIUs impact remote packet capture. Fragmented packets that ingress HIFNs or Cavium NIUs include only the outer header within the fragments. As a result, these traces do not appear to be using IPIP encapsulation. This differs from fragmented packets that ingress the Quad port GiGE and Copper PHY NIUs. These traces include inner and outer headers within the fragments.

Do not run packet-trace simultaneously with other Oracle USM replication features, such as LI, SRS, SIP Monitoring and Trace, and Call Recording. These features may interfere with each other, corrupting each's results.

## Packet Trace Remote

Packet trace remote enables the Oracle USM to mirror traffic between two endpoints, or between itself and a specific endpoint to a user-specified target. To accomplish this, the Oracle USM replicates the packets sent and received, encapsulates them according to RFC 2003, and sends them to a user-configured target. At the target, the user would capture and analyze the packets.

Currently, the Oracle USM supports:

- One configurable trace server (on which you capture/analyze the traffic)
- Sixteen concurrent endpoint traces

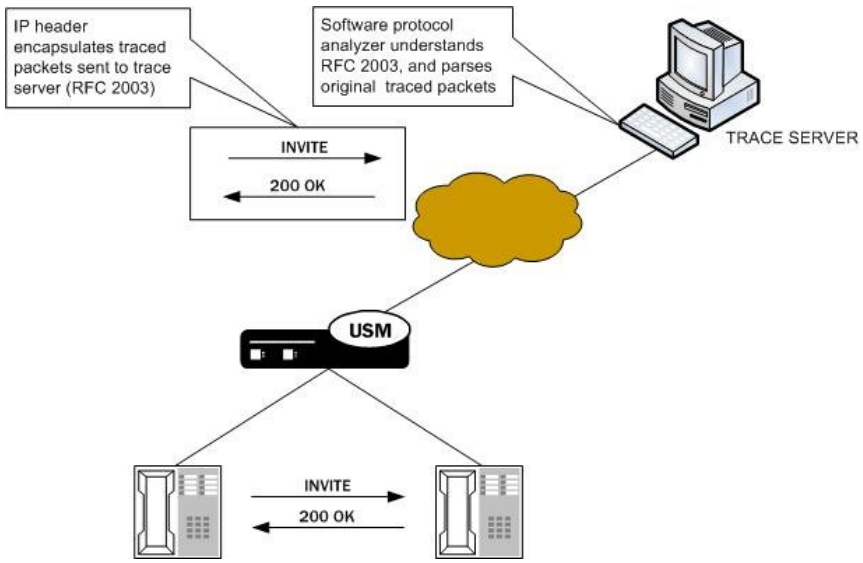
To use this feature, the user configures a **capture-receiver** on the Oracle USM so that it knows where to send the mirrored packets. Once the **capture-receiver** is configured, the user issues the **packet-trace** command to start, stop and specify filters for traces.

You establish a packet trace filter with the following information:

- Network interface—The name of the network interface on the Oracle USM from which you want to trace packets. The user can enter this value as a name or as a name and subport identifier value (name:subportid)

- IP address—IP address of the endpoint to or from which the target traffic goes.
- Local port number—Optional parameter; Layer 4 port number on which the Oracle USM receives and from which it sends; if no port is specified or if it is set to 0, then all ports will be traced
- Remote port number—Optional parameter; Layer 4 port number to which the Oracle USM sends and from which it receives; if no port is specified or if it is set to 0, then all ports will be traced.

The Oracle USM then encapsulates the original packets in accordance with RFC 2003 (*IP Encapsulation within IP*); it adds the requisite headers, and the payload contains the original packet trace with the Layer 2 header removed. Since software protocol analyzers understand RFC 2003, they can easily parse the original traced packets.




It is possible that—for large frames—when the Oracle USM performs the steps to comply with RFC 2003 by adding the requisite header, the resulting packet might exceed Ethernet maximum transmission unit (MTU). This could result in packets being dropped by external network devices, but widespread support for jumbo frames should mitigate this possibility.

If the Oracle USM either receives or transmits IP fragments during a packet trace, it only traces the first fragment. The first fragment is likely to be a maximum-sized Ethernet frame.

The Oracle USM continues to conduct the packet trace and send the replicated information to the trace server until you instruct it to stop. You stop a packet trace with the ACLI **packet-trace remote stop** command. With this command, you can stop either an individual packet trace or all packet traces that the Oracle USM is currently conducting.

# Packet Trace Local

Packet Trace Local enables the Oracle USM to capture traffic between two endpoints, or between itself and a specific endpoint. To accomplish this, the Oracle USM replicates the packets sent and received and saves them to disk in .PCAP format.

The default packet trace filter uses the specified interface to capture both ingress and egress traffic. To specify captured traffic, the user can append the command with a PCAP filter enclosed in quotes. PCAP filter syntax is widely published.

While capturing, the system displays the number of packets it has captured and prevents the user from entering any other ACLI commands from that session. The user terminates the capture by pressing Ctrl+C.

By default, the system saves the .PCAP file in /opt/traces, naming it with the applicable interface name as well as the date and time of the capture. Alternatively, the user can specify file name using the system supports the PCAP filter flags -w.

The system rotates the PCAPs created in this directory by size. The last 25 files are kept and are rotated when they reach 100 MB. If there are capture files in the /opt/traces directory when this command is run, the system prompts the user to remove them before running new captures. If preferred, the user can decline this file deletion.

# Packet Trace Scenarios

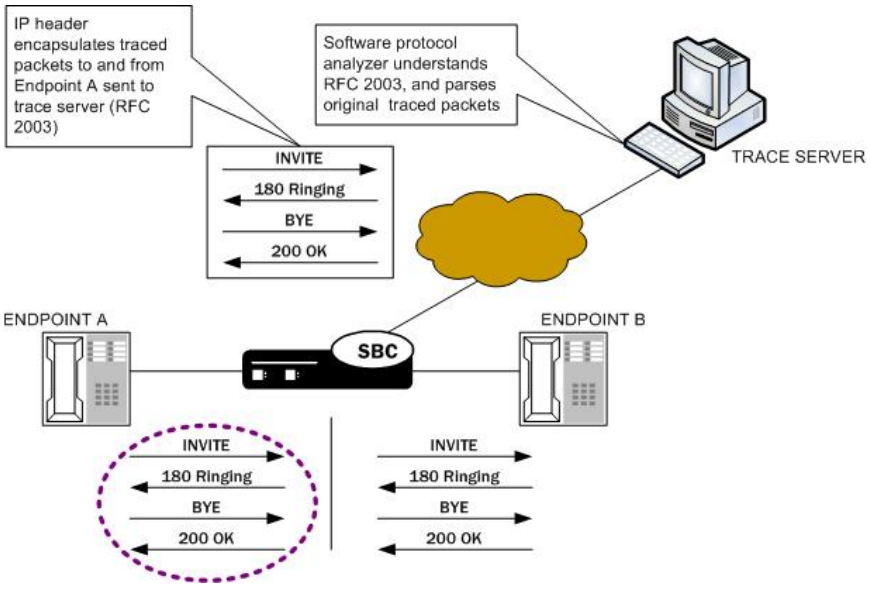
This section describes three possible ways that you might use the packet trace feature. You can examine communications sent to and from one endpoint, sent between two endpoints, or sent between ingress and/or egress Oracle USM interfaces to endpoints.

## Packet Trace for One Endpoint

When you use the packet-trace remote <state> command, the Oracle USM sets up packet tracing for one endpoint. The Oracle USM collects and replicates the packets to and from one endpoint. To enable this kind of trace, you set up one packet trace using the **packet-trace** command.

The commands you carry out for **packet-trace remote** would take the following form:

```
ORACLE# packet-trace remote start F01:0 <IP address of Endpoint A>
```




The commands you carry out for **packet-trace local** would take the following form:

```
ORACLE# packet-trace local F01:0 <"host IP address of Endpoint A">
```

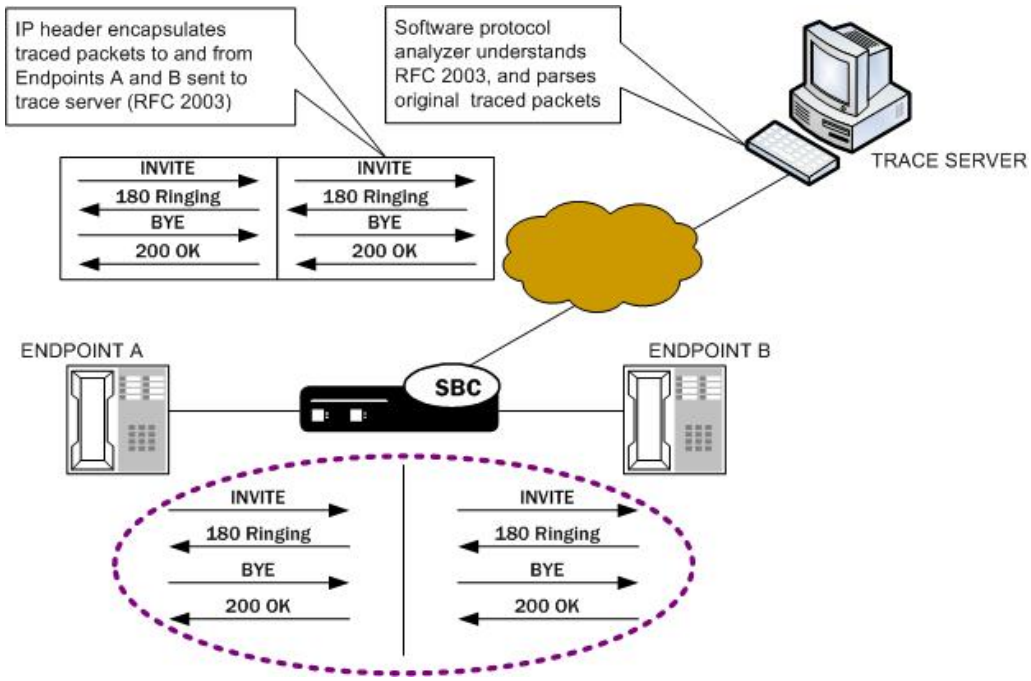
# Packet Trace for Both Call Legs

If you want to trace both sides (both call legs), then you must set up individual traces for each endpoint—meaning that you would initiate two packet traces. The results of the trace will give you the communications both call legs for the communication exchanged between the endpoints you specify.

If you initiate a packet trace for both endpoints that captures both signaling and media, the signaling will be captured as usual. However, RTP will only be traced for the ingress call leg. This is because the Oracle USM performs NAT on the RTP, which means it cannot be captured on the egress call leg.

The commands you carry out for **packet-trace remote** would take the following form:

```
ORACLE# packet-trace remote start F01:0 <IP address of Endpoint A>
ORACLE# packet-trace remote start F02:0 <IP address of Endpoint B>
```




The commands you carry out for **packet-trace local** would take the following form:

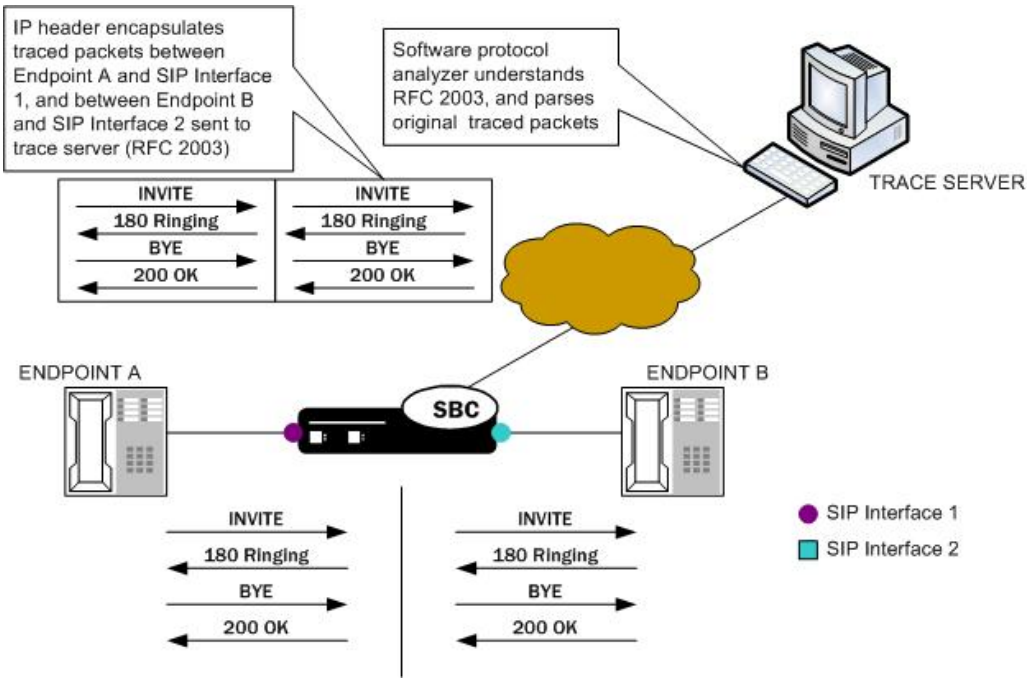
```
ORACLE# packet-trace local F01:0 <"host IP address of Endpoint A">
ORACLE# packet-trace local F02:0 <"host IP address of Endpoint B">
```

# Packet Trace for a Oracle USM Signaling Address

You can perform a packet trace for addresses internal to the Oracle USM; this can be the address, for example, of a SIP interface. Using signaling interface addresses puts the emphasis on the Oracle USM rather than on the endpoints by allowing you to view traffic from specified interfaces.

The commands you carry out for **packet-trace remote** would take the following form:

```
ORACLE# packet-trace remote start F01:0 <IP address of Oracle USM interface1>
ORACLE# packet-trace remote start F02:0 <IP address of Oracle USM interface2>
```




The commands you carry out for **packet-trace local** would take the following form:

```
ORACLE# packet-trace local F01:0 <"host IP address of Oracle USM interface1">
ORACLE# packet-trace local F02:0 <"host IP address of Oracle USM interface2">
```

> **Note:**
>
> The system does not support egress RTP capture with Transcoding NIU

# Running Packet Trace

There are four steps you take when you use the **packet-trace remote** feature. For **packet-trace local**, there are only two.

- Configuring the Oracle USM with the trace server information so that the Oracle USM knows where to send replicated data. (**packet-trace remote** only)
- Setting up the capture filter ip proto 4 in your software protocol analyzer if you only want to see the results of the Oracle USM packet trace(s). (**packet-trace remote** only)
- Starting a packet trace.
- Stopping a packet trace.

This section provides information about how to perform all tasks.

## Configuring a Trace Server

Trace servers only apply to **packet-trace remote**. You need to configure a trace server on the Oracle USM; this is the device to which the Oracle USM sends replicated data. The Oracle USM supports one trace server.

To configure a trace server on your Oracle USM:

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

   ```
   ACMEPACKET# configure terminal
   ```

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

   ```
   ACMEPACKET(configure)# system
   ACMEPACKET(system)#
   ```

3. Enter **capture-receiver** and press Enter.

   ```
   ACMEPACKET(system)# capture-receiver
   ACMEPACKET(capture receiver)#
   ```

4. **state**—Type **enabled** so that you can use the trace server to which you want to send the mirrored packets for calls you are packet tracing. The default is **disabled**. The valid values are:

   - enabled | disabled

     Disable capture receivers you are not actively using for traces to prevent potential service outages caused by the capture's system resource utilization.

5. **address**—Enter the IP address of the trace server; there is no default.
6. **network-interface**—Enter the name and subport of the Oracle USM network interface from which the Oracle USM is to send mirrored packets. Your entry needs to take the form name:subport. The default is **:0**.
7. Save and activate your configuration.

## Starting a Remote Packet Trace

You use the start a remote packet trace by entering the appropriate ACLI command with these pieces of information:

- Network interface (name:subport ID combination)
- IP address to be traced; if you do not enter local and/or remote ports when you start the trace, the Oracle USM will trace all ports
- (Optional) Local UDP/TCP port on which the Oracle USM sends and receives traffic to be traced
- (Optional) Remote UDP/TCP port to which the Oracle USM sends traffic, and from which it receives traffic to be traced; you cannot enter the remote port without specifying a local port
  To start a packet trace with local and remote ports specified:

Note that the system supports packet trace on the Acme Packet 3820, Acme Packet 4500 and Acme Packet 4600, but not on the Acme Packet 1100 and VM Edition.

1. Enter the ACLI **packet-trace remote** command followed by a Space, and the word **start**. After another Space, type in the name and subport ID for the network interface followed by a Space, the IP address to be traced followed by a Space, the local port number followed by a Space, and then optionally the remote port number. Then press Enter.

   ```
   ACMEPACKET# packet-trace remote start core:0 192.168.10.99 5060 5060
   Trace started for 192.168.10.99
   ```

## Stopping a Remote Packet Trace

You stop a remote packet trace by entering the appropriate ACLI command with these pieces of information:

- Network interface (name:subport ID combination)
- IP address to be traced
- (Optional) Local UDP/TCP port on which the Oracle USM sends and receives traffic to be traced
- (Optional) Remote UDP/TCP port to which the Oracle USM sends traffic, and from which it receives traffic to be traced

If the packet trace you want to stop has no entries for local and/or remote ports, then you do not have to specify them.

1. To stop a packet trace with local and remote ports specified, enter the ACLI **packet-trace remote** command followed by a Space, and the word **stop**. After another Space, type in the name and subport ID for the network interface followed by a Space, the IP address to be traced followed by a Space, the local port number followed by a Space, and then optionally the remote port number. Then press Enter.

   ```
   ACMEPACKET# packet-trace remote stop core:0 192.168.10.99 5060 5060
   ```

2. To stop all packet traces on the Oracle USM, enter the ACLI **packet-trace remote** command followed by a Space, and the word **stop**. After another Space, type the word **all** and press Enter.

   ```
   ACMEPACKET# packet-trace remote stop all
   ```

## Starting a Local Packet Trace

You use the start a packet trace by entering the appropriate ACLI command with these pieces of information:

- Network interface (name:subport ID combination)
- (Optional) Enter a tcpdump command line within quotes

Note that the system supports local packet trace on all platforms.

1. Enter the ACLI **packet-trace local** command followed by a Space. Type in the name and subport ID for the network interface followed by a Space, the IP address to be traced followed by a Space, the local port number followed by a Space, and then optionally the remote port number. Then press Enter.

   ```
   ACMEPACKET# packet-trace local s0p0 "host 192.168.12.12"
   Files found in trace directory. Remove [y/n]?: y
   File: /opt/traces/s0p0_0_00001_20150723095442.pcap
   Packets: 5 Packets dropped: 0
   ```

   The ACLI session does not accept use input while the **packet-trace local** command is running.

## Stopping a Local Packet Trace

Type Ctrl-C to stop a local packet trace. This also re-enables the command line session.